

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
27 December 2001 (27.12.2001)

PCT

(10) International Publication Number
WO 01/99372 A2

- (51) International Patent Classification⁷: **H04L 29/00**
- (21) International Application Number: PCT/US01/19332
- (22) International Filing Date: 15 June 2001 (15.06.2001)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
60/212,126 16 June 2000 (16.06.2000) US
09/826,602 5 April 2001 (05.04.2001) US
09/878,093 8 June 2001 (08.06.2001) US
- (71) Applicant: **SECURIFY, INC.** [US/US]; 1157 San Antonio Road, Mountain View, CA 94043 (US).
- (72) Inventors: **COOPER, Geoffrey**; 2542 Webster, Palo Alto, CA 94301 (US). **SHERLOCK, Kieran, Gerard**; 455 Colorado Avenue, Palo Alto, CA 94306 (US). **SHAW, Robert**; 2237 Via Maderos, Los Altos, CA 94024 (US). **VALENTE, Luis, Filipe, Pereira**; 2903 SevysonCourt, Palo Alto, CA 94303 (US).
- (74) Agents: **GLENN, Michael** et al.; Glenn Patent Group, Ste. L., 3475 Edison Way, Menlo Park, CA 94025 (US).
- (81) Designated States (*national*): AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: EFFICIENT EVALUATION OF RULES

(57) Abstract: A method and apparatus uses a proprietary algorithm for organizing network security policy rules in a way that minimizes the number of rules considered when determining the set of rules applicable to a given protocol event.

Best Available Copy

WO 01/99372 A2

Efficient Evaluation of Rules

5

BACKGROUND OF THE INVENTION

10

TECHNICAL FIELD

The invention relates to organizing data for better efficiency at runtime. More particularly, the invention relates to a technique for organizing policy rules to efficiently evaluate protocol events at runtime.

15

DESCRIPTION OF THE PRIOR ART

20 Networked information systems are an essential part of many organizations. Critical systems, services, and information resources all require protection that depends on effective orchestration of a variety of factors: network architecture, security products, site security, administrative procedures, end user responsibility, and more. A network security policy is an explicit plan of
25 how to accomplish this multi-faceted protection, what objectives the plans should meet, and what assets are being protected.

To manage a network, an end user needs to know and understand what is happening on the network. Most security holes come from unexpected, misconfigured, or unauthorized services, for example, from a high-port telnet, a new service added in, a rogue server, and/or a misconfigured workstation. The end user does not know what is the unauthorized network traffic.

Security administrators need tools to help them formulate site security policy and to translate the policy into monitoring and enforcement mechanisms. They need to be sure that the computer enforced policy – often cobbled together from a plethora of disjoint access control mechanisms – matches their enterprise policy, all too often specified in a loose natural language or a set of unwritten principles. This leads to confusion as to why access is being granted or denied to particular resources and may lead to unintentional breaches of security.

In addition to monitoring network system traffic, it is important for network analysts to assess their network's configuration. A discussion on current techniques for network assessment follows below.

A conventional network assessment visit determines the customer network using the following information:

- 1) Network security scanning technology, *e.g.* port or vulnerability scans;
- 2) Customer interviews;

- 3) Inspection of customer log files, perhaps using machine aggregation and filtering; and
- 5 4) Occasionally, inspection of customer log files and network traffic.

As a matter of practicality, the information is typically derived from the first three of these items. Customer log files and network traffic is of a volume so great that it is impractical to examine it in a short assessment visit.

10

The weaknesses such conventional methods are as follows:

Vulnerability Scans

15 Network vulnerability scanners only detect certain types of known vulnerabilities. Such vulnerabilities are generally not detected directly, but are inferred based on host responses to a series of network packets sent to hosts by the scanner. This process does not directly ensure that data traffic on the subject network matches expectations, either explicit or implicit.

20 Network vulnerability scanners cannot see a host if it does not respond to packets. A host that is only a source of network packets, such as, for example, a rogue router, is not visible to a scanner. Hosts which are turned off or otherwise temporarily disconnected, such as, for example, workstations and laptops, are often missed by vulnerability scanners. This problem is
25 compounded by the fact that scans are often scheduled for non-work hours in

order to alleviate customer fears that the scans will somehow impact production systems and organizational mission.

5 Network scanners typically return a large volume of vulnerability information, based on all possible configured elements in a network. The scanner tools cannot currently interpret those vulnerabilities in light of business requirements which the subject systems are intended to support, or even for the specific network architecture of which those systems are a part. The scan results must be reviewed manually by a security analyst, who applies a
10 knowledge of the business requirements and network architecture to an interpretation of those results. Such manual process is error-prone because the volume is so great that problems may be overlooked.

Another problem is that the scan derives only vulnerabilities, not network
15 usage patterns. Therefore, the scan cannot detect security problems that are attributable to human behavior, but only those scans that result from misconfigured systems and/or systems which have documented design problems.

20 Network scanners cannot diagnose incorrect client usage of software. For example, network scanners cannot detect whether web servers are being used with invalid ciphersuites, whether 40-bit browsers are in use, and whether a given telnet port is accessed only by a management station.

25 Network scanners must be targeted to particular subnets. If a customer has forgotten to mention a subnet, the scanner does not notice it.

Customer Interviews

Customers may not provide the network analyst complete or accurate
5 information, either because the customer forgot details, because the
information is not known to the customer, or because the customer does not
understand the importance of giving the information to the analyst.

Customer interviews at best can provide descriptions of overt usage of subject
10 systems, and generally not covert usage. Often, formal policies of the
organization are not even documented, much less promulgated, audited and
enforced.

Hidden agendas, office politics, and other factors also can affect the success
15 of the interview process.

Host Inspection

Inspecting host configuration files is a time consuming, manual process that is
subject to human error. In the assessment of any large network, it is
20 impractical to include an inspection of the configurations for more than a few
critical systems.

Once again, inspection of host configurations does not reveal completely
intended usage of the subject systems. The configurations must be analyzed
25 within the context of the business requirements and overall security

environment of the organization. This manual process is very human dependent and prone to error.

Log File Inspection

- 5 Log file inspection can provide great insight into the workings of network components. Machine-based aggregation and filtering systems can speed this process. However, logs provide only a components' own view of its status. If a component is misconfigured, the log data from the component cannot be trusted. Log data may also be subject to modification by an
10 attacker who has penetrated the machine and is seeking to mask his presence.

In addition, because log aggregation systems work in cooperation with the components that generate the information, they require configuration changes
15 to every component that they examine. Also, they are unable to detect when a component is added to the system.

Such techniques of performing network assessments generally are limited in their ability to determine actual security threats to information systems.
20 Generally, they represent the state of the art and are indicative of best practices within the security community today.

A way to reduce or eliminate the confusion described above is by providing a user-friendly and, yet, rigorous way of specifying security policy, as well as
25 providing tools for monitoring and enforcing the security policy.

Blaze, Feigenbaum, and Lacy (BFL), *Decentralized Trust Management*, Proc. IEEE Conference on Security and Privacy (1996), used the term *trust management* to refer to a problem of deciding whether requested actions, supported by credentials, conform to policies. In other words, it deals with the questions of who, how, and what. *Who* (the principals, for example, people, computers and organizations) can access *what* (the resources being sought) and *how* (the actions performed against the target resources).

Mansouri-Samani, *et al*, *GEM: A Generalized Monitoring Language for Distributed Systems*, Distributed Systems Engineering, vol.4, no. 2 96-108 (June 1997) discloses a generalized-event monitoring notation that permits user-specified filtering and composition scripts to be dynamically loaded into distributed-event monitoring components. GEM uses "scheduled time events and default or user-defined detection windows" to cope with "variable communication delay problems." The GEM event monitoring system is used "to detect complex event sequences and to convert these into simple events" that trigger management actions. The event monitors have been restricted to performing "very simple activities related to triggering or notifying events."

J. A. Grompone, *A Declarative Language for the Configuration of Exchanges*, Telecommunications Journal, vol. 56, no.1 (Jan. 1989) discloses the design and implementation of a high-level language, LEP, to define the routing and customizing of rules of a telex exchange. The routing concepts are basic and few in number. Each of the physical communication paths is called a line. The lines are arranged in groups. The purpose of the LEP language is to provide a comprehensive definition of all lines of an exchange, the

arrangement of these lines in groups and the physical attributes of the groups. All groups taken together comprise all the lines without any lines missing or being repeated. A group is an ordered set of lines. The LEP term "access" is used to denote whether lines are permitted or forbidden to access other lines or services. Routing, a basic objective of an LEP program, is a way of associating sets of compiled codes with destinations, done through a sequence of elementary declarations. LEP also defines the possible destinations of a call. One of the main design concepts was to use a very simple structure for the declarations for even users unfamiliar with computer programming.

The LEP language cannot thread together multiple protocol layers of a network event. The LEP language lacks the sophistication in terms of richer expressions to allow a set of policy rules affecting different networking protocols to be applied to a complex protocol interaction between two communicating parties, and to security policy for an entire network. The LEP language does not suggest defining allowed traffic patterns and handling those events that deviate from those patterns.

Plasek, *et al*, *Statistical Database Query Using Random Sampling Of Records*, U.S. Patent 5,878,426, discloses a method for obtaining decision support query results from a database table having multiple records. An attribute of the database table is sampled, which results in a collection of sampled data. The sampled data represents some percentage of all of the data corresponding to that attribute in the database table. The data associated with the attribute includes multiple data classes, and the sampled

data is separated or partitioned into these data classes. A database query is applied to the sampled data rather than to all of the data corresponding to that attribute in the database table.

- 5 Plasek, *et al*, also discloses a method to obtain decision support query results from a database table where all of the data associated with a particular database attribute is grouped into various data classes. Each of the data classes is individually randomly sampled to obtain a corresponding number of class data samples. Each of the class data samples is then queried, which
10 can include executing aggregation functions on each of the class data samples.

Plasek, *et al*, also discloses a method for providing result approximations in database queries.

15

- Plasek, *et al*, does not disclose nor suggest providing a method to select a most specific and applicable result or policy rule. Plasek, *et al*, does not disclose nor suggest providing a method to rank data and does not order data in a database beyond partitioning data into classes and thereafter randomly
20 sampling each data class such that database queries are applied to each of the samples.

- Plasek, *et al*, does not disclose nor suggest providing a method to thread protocol layers of a network event together to provide a result to the network
25 event.

Chow, et al, System, Method, and Program for Extending a SQL Compiler for Handling Control Statements Packaged with SQL Query Statements, U.S.

Patent No. 5,875,334 (February 23, 1999) discloses an integrated compiler for compiling SQL3 control statements having procedural, i.e., control,

5 information packaged together with query, i.e., non-procedural, statements. A

query extractor contained within the parser extracts the query statement from the control statement leaving a control skeleton. The query statement is processed as usual through a query compiler for generating executable plans with the exception that the name resolution function for resolving variables is

10 modified for looking up local variables. This modification takes into account the mapping of local and host variables to create a unification of local and host variables. The control skeleton is processed through a control analyzer

which generates a representation of the control flow and a scope and symbol table. The control analyzer also unifies the local and host variables. A plan

15 synthesizer then takes as input the control flow information, symbol tables, and individual executable plans for the query statements and generates a meta-plan comprising a merger of a top level plan for the control skeleton and sub-plans representing the executable plans of the query statement.

20 Chow, *et al*, does not disclose nor suggest a ranking method or an ordering method to handle a set of rules to be applied to a complex protocol interaction between two communicating parties.

Nor does Chow, *et al*, disclose or suggest a method whereby to thread
25 protocol layers of a network event together to provide a rule applicable to the network event.

V. Paxson, *Bro: A System for Detecting Network Intruders in Real-Time*, Network Research Group, Lawrence Berkeley National Laboratory, Berkeley, CA, LBNL-41197 (Jan. 1998) discloses a stand-alone system for detecting
5 network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits. The system comprises a "policy script interpreter" that interprets event handlers written in a specialized language used to express a site's security policy. The specialized language is C-style because it comprises, for example, C-style data types and constants,
10 operators, and block statements and is procedural. Bro comprises first-class values and aggregate types such as record and table, used to specify a security policy.

However, Paxson does not disclose nor suggest providing a sophisticated
15 ranking method to rank policy rules according to the specificity of the initiator and target communicating hosts and to select a most applicable rule in an efficient manner. Paxson does not disclose nor suggest providing a method to thread protocol layers of a network event together to provide a result to the entire network event.

20 It would be advantageous to reduce or eliminate the confusion described herein above by providing a user-friendly and, yet, rigorous way of specifying security policy, as well as providing tools for monitoring and enforcing the security policy.

25

It would be advantageous to have a trust manager that takes as its input a security policy defined as a set of *policy rules* (statements about trust) and a set of *credentials* (statements about principals), such that it is capable of processing requests for *trust decisions*, i.e. evaluating compliance with the
5 policy.

It would be advantageous for the trust manager to have a unified view of an interaction between two principals across a stack of protocol layers, each governed by discreet policy rules, and to apply a final trust decision based on
10 which of these policy rules better fits the entire interaction. For example, using HTTPS to access a secure web page involves an interaction between two network addressable machines (at the TCP/IP level), an interaction between a cryptographically authenticated server and, possibly, a cryptographically authenticated client (at the SSL level), and an interaction
15 between a Web browser (possibly with its own authentication credentials) and a web server, resulting in the retrieval of a web page (at the HTTP level).

It would be advantageous to have a *policy definition language* as well as well-designed algorithms to support monitoring and auditing network activity, in
20 addition to traditional access/deny authorization decisions. For example, a policy rule might instruct a monitoring Agent to log all traffic between two computers or to decrypt a secure channel between two users.

It would be advantageous to provide a system that comprises a passive
25 monitor of network traffic that does not need to be installed on target hosts or integrated into existing applications.

It would be advantageous to provide a system that uses a sophisticated algorithm for determining which policy rules take precedence over others.

- 5 It would be advantageous to provide a policy language that allows a set of policy rules affecting different networking protocols to be applied to a complex protocol interaction between two communicating parties. Also, it would be advantageous to use the policy language to express security policy for an entire network.

10

It would be advantageous to provide a system, unlike current Intrusion Detection Systems (IDS) which only look for signatures of known attacks, focusing on defining allowed traffic patterns and determining how to handle events that deviate from those patterns.

15

It would be advantageous for a network policy to provide the definition of normal traffic on the network.

20

It would be advantageous to provide a monitoring mechanism that lets an end user determine and understand traffic and/or activity on a network.

25

It would be advantageous to provide methods and system that, when given known network characteristics, thereby spots intruder access, and track changes to a network.

It would be advantageous to provide a policy generator tool that assists an end user in generating security policy for a network.

It would be advantageous to provide a tool that automatically converts a
5 network security policy into English language representation.

It would be advantageous to provide a tool that allows an end user to query network traffic data.

10 It would be advantageous to provide a technique for transmitting an event description of network traffic from a source file or data stream to a target destination, such as a network policy engine.

15

SUMMARY OF THE INVENTION

The invention is a network security policy method and apparatus that uses a proprietary algorithm for organizing network security policy rules in a way that minimizes the number of rules considered when determining the set of rules
20 applicable to a given protocol event.

The invention can be a component of a network security policy monitoring system and method that comprises supportive features, algorithms, and tools. The monitoring system is ideally suited for network and security assessments
25 or long-term monitoring where real network traffic is analyzed to identify abnormal traffic patterns, system vulnerabilities, and incorrect configuration of

computer systems on the network. The monitoring system listens on a network, logs events, and takes action, all in accordance with a rule based system-wide policy. The monitoring system provides a technique that is able to incorporate external sources of event information, such as are generated in log files of other network components. The inventive technique of the monitoring system gets protocol information, which can make it more meaningful to a network administrator. It sends data upstream to an event log and interprets the data. It listens to secure protocols and can identify encryption quality of service parameters. It extracts basic security parameters, such as, for example, network events, and passes them to a policy manager component.

The policy manager component implements system-wide policies, based on monitored system or enterprise traffic. The policy manager component provides a trust manager that takes as its input a security policy defined as a set of policy rules and a set of credentials, and that is capable of processing requests for trust decisions, *i.e.* evaluating compliance with the policy. Unlike other trust management systems, the monitoring system is designed to be a passive monitor of network traffic. As such, it need not be installed on target hosts or integrated into existing applications.

Two key aspects of the policy manager component are provided. One aspect is a unified view of the interaction between two principals across a stack of protocol areas, each area covered by discrete policy rules. The final trust decision applied is based on policy rules that better fit the entire interaction. The second aspect comprises the policy manager's policy definition language

that supports the monitoring and auditing of a network's activity in addition to traditional access/denial authorization decisions.

The policy definition language is described in A Declarative Language for Specifying A Security, U.S. patent application serial number 09/479,781, (01/07/00), also included in section, A Declarative Language for Specifying a Security Policy herein below. The policy definition language is discussed herein to the extent necessary to explain such language to those skilled in the art in connection with the invention and the monitoring system disclosed herein. The declarative language system comprises a language as a tool for expressing network security policy in a formalized way. It allows the specification of security policy across a wide variety of networking layers and protocols. Using the language, a security administrator assigns a disposition to each and every network event that can occur in a data communications network. The event's disposition determines whether the event is allowed, *i.e.* conforms to the specified policy or disallowed and what action, if any, should be taken by a system monitor in response to that event. Possible actions include, for example, logging the information into a database, notifying a human operator, and disrupting the offending network traffic. Further details of the policy definition language can be found in the patent application cited herein above.

Unlike Intrusion Detection Systems (IDS) systems, which look for the signatures of known attacks, the monitoring system herein is focused on defining allowed traffic patterns and how to handle events that deviate from those patterns.

The monitoring system comprises, but is not limited to six major features and tools. The first feature discussed is auto-conversion of policy language, whereby policy language is converted to an English language representation.

- 5 Next, an algorithm for efficient rule evaluation is provided. Then, a credential/assertion optimization technique is provided. A policy generator tool is provided. An embodiment in which the monitoring system is used as an assessment tool is provided. Finally, a technique for secure sensitive event extraction from protocol monitoring is provided.

10

The invention comprises a declarative language system and comprises a language as a tool for expressing network security policy in a formalized way. It allows the specification of security policy across a wide variety of networking layers and protocols. Using the language, a security administrator assigns a
15 disposition to each and every network event that can occur in a data communications network. The event's disposition determines whether the event is allowed, *i.e.* conforms to the specified policy, or disallowed and what action, if any, should be taken by a system monitor in response to that event. Possible actions include, for example, logging the information into a database,
20 notifying a human operator, and disrupting the offending network traffic.

The language is implemented by a policy engine, a component also of a security policy monitoring (SPM) system. The SPM system, also referred to herein as the policy monitoring system, is ideally suited for network and
25 security assessments where real network traffic is analyzed in order to identify

abnormal traffic patterns, system vulnerabilities and incorrect configuration of computer systems on the network.

Unlike other trust management systems, the SPM is designed to be a passive
5 monitor of network traffic. As such, it need not be installed on target hosts or integrated into existing applications.

The invention provides a simple and intuitive model for expressing and applying security policies. The language is much richer in terms of what it can
10 express than languages used in firewalls and routers. It uses a sophisticated algorithm for determining which policy rules take precedence over others, a process that in other systems is completely manual.

Unlike existing firewalls and routers, the invention allows a set of policy rules
15 affecting different networking protocols to be applied as a whole to a complex protocol interaction between two communicating parties. Furthermore, networking equipment typically handles only policy related to the network traffic that flows through it. Using the invention herein one can express the security policy for an entire network.

20

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1a is a schematic diagram of components of the system according to the invention;

25

Fig. 1b is a schematic diagram of components of the system according to the invention;

Fig. 2 is a high level workflow flow diagram according to the invention;

5

Fig. 3 is an example of a policy wizard dialog box according to the invention;

Fig. 4a is an example of a policy wizard dialog box according to the invention;

10 Fig. 4b is an example of a policy wizard dialog box according to the invention;

Fig. 5 is an example of a policy monitor dialog box according to the invention;

Fig. 6 is an example of a query tool dialog box according to the invention;

15

Fig. 7 is an example of a query tool dialog box according to the invention;

Fig. 8 is an example of a query tool dialog box according to the invention;

20 Fig. 9 is an example of a query tool dialog box according to the invention;

Fig. 10a is an example of a policy wizard dialog box according to the invention;

25 Fig. 10b is an example of a policy wizard dialog box according to the invention;

Fig. 10c is an example of a policy wizard dialog box according to the invention;

- 5 Fig. 11 shows a high-level view of an example network according to the invention;

Fig. 12 shows an algorithm according to the invention;

- 10 Fig. 13 shows a flow diagram according to the invention;

Fig. 14 shows an algorithm according to the invention;

Fig. 15 shows a high level schematic diagram according to the invention;

15

Fig. 16 shows a schematic diagram of process flow according to the invention;

Fig. 17 is a block schematic diagram according to the invention;

- 20 Fig. 18 is a high level flow diagram of the preferred output section according to the invention;

Fig. 19 shows a schematic diagram according to the invention;

- 25 Fig. 20 is an example of a dashboard according to the invention;

Fig. 21 shows an example of a tear off console according to the invention;

Fig. 22 shows an example of an events summary view according to the invention;

5

Fig. 23 shows an example of a conformance event details page according to the invention;

Fig. 24 shows an example of a protocol event details page according to the invention;

10

Fig. 25 shows an example of an events summary page containing a pop up description according to the invention;

Fig. 26 shows an example of an events summary page containing a pop up description according to the invention;

15

Fig. 27 shows an example of a conformance event details page containing a pop up description according to the invention;

20

Fig. 28 shows an example of an alert details page according to the invention;

Fig. 29 shows an example of a violators chart and table page according to the invention;

25

Fig. 30 shows an example of a targets chart and table page according to the invention;

Fig. 31 shows an example of an advanced search dialog box according to the
5 invention;

Fig. 32 shows an example of a link to the advanced search dialog box according to the invention;

10 Fig. 33 is a schematic diagram showing the relationship of elements of the Policy Monitoring System, according to the invention;

Fig. 34 is a schematic diagram of a protocol event according to the invention;

15 Fig. 35 is a schematic diagram of a disposition according to the invention;

Fig. 36 is a schematic diagram of communicating parties according to the invention;

20 Fig. 37a is a schematic diagram of a network event, comprising protocol events at different protocol layers, having an associated network event disposition according to the invention; and

Fig. 37b is an algorithm showing protocol events at different protocol layers
25 resulting in pending rules with or without immediate outcomes and, finally, a final disposition for the network event.

DETAILED DESCRIPTION OF THE INVENTION

The invention is a security policy monitoring system and its supportive
5 features, algorithms, and tools. It is ideally suited for network and security
assessments where real network traffic is analyzed in order to identify
abnormal traffic patterns, system vulnerabilities, and incorrect configuration of
computer systems on the network. The system listens on a network, logs
events, and takes action, all in accordance with a rule based system-wide
10 policy. The system is able to incorporate external sources of event
information, such as are generated in log files of other network components.
The system gets protocol information, which can make it more meaningful to a
network administrator. The system sends data upstream to an event log and
interprets the data. The system listens to secure protocols and can decrypt a
15 session if a key escrow facility is available. The system extracts basic
security parameters, such as, for example, network events, and passes them
to a policy manager component.

Efficient Evaluation of Rules

20

An important part of understanding the invention is understanding network
security terminology for policy monitoring. See Table A below.

Table A

25

Terminology

Network Event: One complete transaction on the network, such as a FTP connection or a HTTPS transaction. Each network event has several component *protocol events*.

5

Protocol Event: A transaction at one protocol level. For example, a network event that represents an FTP connection has protocol events representing an IP association, a TCP connection, an FTP control connection, and several FTP control commands.

10

Initiator, Target: The endpoints of a network event or protocol event.

Credential: An identification of the initiator or target of a protocol event at a particular protocol level. For lower-level protocols, credentials are, for example, IP addresses or UDP port numbers. For higher level protocols, credentials are, for example, user names, file names, or public key certificates.

15

Association: A placeholder for a transaction run over a datagram-based protocol such as IP, ICMP or UDP. The invention herein constructs an association to collect a conversation between two hosts, or processes in the case of UDP. It is noted that when the invention misses any data packets between the two communicating computers, it might not be able to determine the initiator and the target of the association.

20

Associative array: A list of value pairs where each associative array entry is indexed by the first element of its value pair, which is called the key. Keys are stored in a hash table to make lookups efficient irrespective of the size of the associative array.

25

Rule: A policy rule governs a specific interaction, or set of interactions, between two communicating entities. The invention evaluates policy rules against protocol events to determine if the latter conform to the active security policy.

30

Disposition: The policy definition of what action or state change needs to take place in response to a network event.

5 *Policy Domain:* A top level segmentation of a network, roughly akin to a cloud-like object in a network diagram, which hides internal detail. Within the policy domain communities of hosts provide or access services. One community of hosts defines the limits of the domain.

10 *Monitoring Point:* A point within a policy domain where it will be possible to plug a machine into the network in order to collect packet data.

Communities of Hosts: A mechanism for grouping hosts that have a similar function, e.g. all web servers or all NT workstations.

15 *Perimeter Element:* A hardware device that allows access to and from communities of hosts outside a policy domain. Examples of perimeter elements are firewalls and routers.

20 *Policy Language:* A policy language is used to create a formal specification of a network security policy. The preferred embodiment of the invention incorporates the policy definition language of U.S. patent application number 09/479,781, filed 01/07/00, entitled, "A Declarative Language for Specifying A Security Policy." It defines first class objects such as rules, credentials and dispositions. It is based on
25 s-expressions, which are LISP-like parenthesized expressions.

Rogue server: A machine introduced to a network that is not authorized to be on that network.

Rogue router: An unauthorized router that is added to a network, providing an alternate path into the network. Typically occurs through misconfiguration of switches or dialup connections.

5 *Real-time monitoring:* Reading packet data off a network and processing it to events in a stream, so that an event appearing in the network causes a corresponding event in the stream a short time later.

DLL: Any kind of a dynamically linked library

10

System Overview

The preferred embodiment of the invention translates traffic on the network into protocol events that are themselves combined into network events. As
15 protocol events are detected, they are compared against a policy. The policy specifies a disposition of the network event, as defined by the observed series of protocol events. Information about the protocol events, the network event and its disposition is stored in a database. This database of network traffic information can be mined for policy violations.

20

This preferred embodiment of the invention is described with reference to Fig. 1a. Fig. 1a is a schematic diagram of components of the system according to the invention. The system comprises a policy monitoring component 100 that takes as input a policy file 105 that has been generated using a policy
25 generator wizard 110 or other means, and a file containing network packet dump data 115 that has been collected from an observed network 125 by a packet capture 126, or that has been processed by a protocol monitor

processor 127. The system can also process packet event data from the observed network 125 in a continuous real-time mode, without first storing packet data to a file.

5 The policy monitoring component 100 comprises a policy manager component 106 that itself comprises a parser 101 for parsing the policy file 105, a policy engine 102 for assigning policy dispositions to network events, and a logger 103 for determining how to log the information processed by the policy engine 102, according to an input logging policy 130. It also
10 comprises a database 104 for storing synthesized information of the packet dump's 115 conformance to the specified policy 105 performed by the policy engine 102, where it can be mined with a query tool 135. It also comprises a report script component 160 for querying the database 104 and creating reports 161, and an alarm script component 155, for generating alarms based
15 on the severity of the disposition assigned to network events.

An equally preferred embodiment of the invention also comprises a parser tool 150 that takes the policy specification file 105 as input and automatically generates an English description of the policy 151 for the end user. The
20 parser tool 150 is optional.

An equally preferred embodiment of the invention also provides a secure Web server feature 162 for the end user to review reports from the end user's host computer 163. The secure Web server feature 162 comprises the Web server
25 164 and a report database 165 that hosts the reports 161 generated using the report script 160. The Web server feature 162 is optional.

An equally preferred embodiment of the invention provides secure management connections (141, 142) and a secure management host 140 for managing the policy monitoring component 100 and the combination of the
5 network monitoring components 128, respectively.

Fig. 1b shows a simpler embodiment of the invention, wherein the parser tool 150 and the secure Web server feature 162 are omitted.

10 The default action of the policy engine 102 is that it denies all traffic. The policy 105 opens holes in this denial to allow permitted traffic to flow. Although the policy engine 102 assigns a single disposition to an entire network event, the protocol events are significant. As network data 115 arrives, the policy engine 102 interprets protocols and generates updates of
15 protocol event information. The policy 105 is consulted as each new piece of information arrives, so that the earliest determination of disposition is reached. For example, if the policy 105 states that a given IP address may not communicate with another IP address, the policy 105 can generate a disposition immediately upon receiving the first packet 115 of the network
20 event.

To aid policies in early determination of disposition, the policy language divides dispositions into immediate and final. An immediate disposition fires immediately, *i.e.* its value becomes associated with the network event right
25 away. A final disposition sets a bookmark to itself as the latest and best disposition. When all protocol events are processed without an immediate

disposition, the last bookmark set is the disposition that is applied to that network event. Immediate dispositions are designed to generate early results and to allow policy writers to issue a definitive disposition for the network event based on the information received up to that point. Final dispositions

- 5 allow for the possibility that a better disposition might be determined later on. In other words, they allow the policy engine 102 to make a more informed decision based on additional protocol events that might be received as the network event progresses.

10 Overview of the Components

An overview of main components of the preferred embodiment of the invention is discussed below with reference to Fig. 1.

Policy Generator

- 15 The preferred embodiment of the policy generator component 110, also referred to as policy wizard, is a program that makes an end user readily able to generate a first-pass policy for a new site. Policy information is input into a set of dialog boxes and a policy is generated. The wizard enables the end user to generate policy based on what can be considered gross
- 20 characteristics of a network at the IP level, such as, for example, policy domains, communities of hosts, servers, subnets and firewalls, as well as at the UDP/TCP service level. For example, such network characteristics can comprise communities of hosts that can access certain services on server hosts.

Once a policy has been generated with the wizard, it is output in the policy specification language 105 so that it may be directly processed by the policy monitor component 100. The policy wizard 110 is also able to save files at the wizard level, *i.e.* such that the policy may be refined in the wizard and re-generated.

Policy Monitor

The policy monitoring component 100 comprises a suitable user interface, such as an MFC-based front end or a command line interface, and the policy manager 106. The policy manager 106 performs the actual examination of a sequence of event updates stored in a file or transmitted in a continuous stream 115 in the context of a policy specification 105 and signals the adherence to the policy via records written to the database 104.

15 Network Monitor

The network monitor component 127 provides the following capabilities:

- Streams-based interpretation of packet dump data 126 in, for example, DMP format; and

20

- Packet- and connection-based textual logging of protocol information. Logging is selectable by protocol and may be enabled only for one or more connections. In another embodiment of the invention, the network monitor 127 can perform serialization of event data. That is, the network monitor 106 can process a packet capture file 126 into a series of event updates that contain only the salient security details for processing by the policy

25

monitor 100. The resulting file is significantly smaller than the original, for example, approximately 1/20th to 1/100th the size of the original. It is also possible for sensitive data, such as passwords and documents, to be removed from the file. However, it should be appreciated that the original
5 packet capture file is needed to perform full analysis.

In another embodiment of the invention, the network monitor 127 can read packet data directly from observed network 125, generating a continuous stream of event updates for the policy monitor 100. This stream operates in
10 real-time so that the policy monitor 100 processes events shortly after they happen on observed network 125.

It should be noted that the network monitor 127 can be used as a standalone tool, but typically is invoked from within the policy monitor component 100 and
15 the query tool 135 in normal operation of the invention.

It should also be noted that the network monitor and the policy monitor may run on the same machine.

20 For a more detailed discussion on the internals of the network monitor, refer to the section, below entitled "Network Monitor Internals Descriptions."

Query Tool

The query tool 135 allows the end user to view the data that has been stored
25 in the database 104 by the policy manager 106.

Policy Compiler

The policy compiler performs syntactic and semantic checking of a policy specification. Upon successful compilation the compiler as controlled by runtime arguments, may:

5

- Generate a DLL containing a compilation of credential and condition verification code; and
- Generate a pseudo-english report that summarizes the policy.

10

It should be appreciated that it is not necessary to run the compiler because the policy monitor component automatically compiles and installs policy from the policy specification file.

15 **Platform**

The policy generator 110 runs on a Windows NT or Unix machine, while the policy monitor 100, and the network monitor 127 run on Linux machine(s). It should be appreciated that these components can run equally well on other suitable operating systems. In addition to policy and network monitoring software, the following software components are also installed on the

20

- Microsoft Visual C++ 6.0;

25

- Sybase ASE 11.9.2; and

- NT NDIS packet drivers and Windump 2.0.

It should be appreciated that these components can run equally well on other compilers, databases, and packet monitoring systems.

5

Policy Files

There are two file types that are used within the invention's environment, and are described below in Table B.

10

Table B

File Type	Suffix	Description
Policy wizard File	.spw	Intermediate file used by the policy wizard to store policy information between invocations.
Policy monitor File	.spm	Output file generated by the policy wizard and used as the policy input into the policy monitor. Contains a description of the policy in the policy language.

The preferred embodiment of the invention incorporates a high level workflow method for developing policy, as follows:

15

- 1) Creating an initial policy using the policy generator tool;
- 2) Uploading the policy file to a remote machine;

3) During the initial policy development phase, running the network monitor to collect traffic, and the policy monitor to analyze traffic separately, as follows:

a) Running the network monitor and specifying an output file of the collected traffic, and possibly specifying via parameter a limit to the number of packets captured, *e.g.* 50,000;

b) Running the policy monitor to analyze traffic collected by specifying the file containing the collected traffic;

4) Examining the output of the policy monitor run by querying the database using the query tool;

5) Modifying the policy as needed using the policy generator tool; and

6) Repeating steps 2 through 5 until a comprehensive desired policy is defined. At this point the end user may start monitoring network traffic on a continuous basis, and using generated reports as input for further policy refinement.

High Level Workflow Example

The high level workflow described above can be illustrated further by understanding an example, as follows. System components of the invention are referenced using Fig. 1. Screen interactions are described with reference to the preferred embodiment of the invention. Other screen displays with similar function might equally well embody the invention.

Referring to Fig. 2, an initial policy is generated (201). Often the initial policy is created from corporate network policy, in whatever form that may take, and a network topology diagram. For the sake of this example, it is assumed that
5 the policy wizard 110 was used to generate an initial, simple policy 105.

Next, compliance of current network traffic to this initial policy is monitored (202). Such monitoring is achieved by collecting packet information off the network and running such data 115 against the initial policy 105 using the
10 policy monitor 100.

Then the query tool 135 is used to data-mine output network event data from the database 104, using the mined data to check for traffic that is not consistent with the policy 105, and reporting the results (203).
15

Once anomalies have been found, the next step is to work out where the problem lies. The problem could be network equipment is misconfigured and needs to be corrected (203); otherwise acceptable behavior is not covered currently by the policy specification file the file needs to be corrected (204); or,
20 otherwise acceptable behavior is not covered currently by the corporate policy and the corporate policy needs to be corrected (205). In the case of this example, it is assumed that the policy specification 105 is incomplete and an end user needs to add a new rule to permit the observed traffic pattern.

25 **Generate a Policy Specification File From a Wizard Policy**

The end user starts the policy generator tool, or wizard 110, by double clicking on a policy wizard shortcut on the end user's desktop. In the preferred embodiment, a window such as depicted in Fig. 3 opens.

- 5 In this example, the end user has opened a file, c:\spm\quickstart\null.spw, through the File->Open menu item 301. This file contains a very simple policy that defines a single policy domain defined by a 10.0.0.0/8 subnet mask. Rules within this policy deny essentially all traffic.
- 10 The end user chooses to compile the policy, whereby the dialog box in Fig. 4 opens. The end user presses the "Process Policy" button 401 and a file named null.spw in the output file entry field 402 is generated and saved.

- Fig. 4b shows the dialog box in Fig. 4a with printed results from the compile process in a text window 403.
- 15

File Running Policy Monitor Over Canned Data

- The end user starts the policy monitor 100 by double clicking on a policy monitor shortcut on the desktop. In the preferred embodiment, a window such as depicted in Fig. 5 opens.
- 20

- The end user ensures that the "Input Dump File" entry field 501 points to a data dump file, here qs.dmp, and that the "Policy" entry field 502 points to the null.spw (monitor) file that the end user generated above. The "Monitoring Point" entry field 503 is derived from a policy domain name "Intranet" that is present in the null.spw (wizard) file.
- 25

The end user ensures database connectivity information is set correctly. The ODBC entry field 504 with entry "sybase" points to a Sybase database running on a local machine. The username "policy" 505 with some password, shown as "*****" 506 have been preinstalled.

The end user presses the Run button 507 and the .dmp file is processed through the policy specification file 105 placing the output data into the database 104.

10

Look at the Results Using Query Tool

The end user starts the query tool 135 by double clicking on a query tool shortcut on the desktop. In the preferred embodiment, a window such as depicted in Fig. 6 opens.

15

The end user presses a "Network Events" button 601 and the dialog box depicted in Fig. 7 appears. Fig. 7 is a dialog box that allows the end user to enter login information for the database 104.

20 Here, the end user enters the same username and password as was used in policy monitor 100 and connects to a database 104 named Policy on localhost.

When connected, the screen shown in Fig. 8 appears. Fig. 8 is a dialog box that allows the user to select which processed network data to view from database 104. The topmost entry in the "Execution Run" pull-down contains

most recent data was added to the database 104. In this case it is current processing of the qs.dmp file. The end user presses the "Query" button and network event information for this run is retrieved from the database 104 and shown in as in Fig. 9.

5

Fig. 9 shows a queried rule view dialog box according to the preferred embodiment of the invention. Fig. 9 shows that the null.spw policy has denied all traffic. The network events having disposition `Udp_Access_Denied` represent DNS lookups from an internal host (10.5.63.143) to another internal host (10.5.63.6). It is assumed for this example that this is traffic conforming to policy, and therefore the end user adds a rule to the policy to permit this event.

Add a New Rule Using The Wizard

15 The end user returns to the policy wizard main window and presses the "Edit Rules" button which opens a dialog box as shown in Fig. 10a. Fig. 10a shows a dialog box for generating a new rule according to the invention. The end user selects the "Intranet" domain from the "Policy Domain" pull-down to add a rule for our Intranet domain. The end user types a rule name, such as

20 `Internal_Dns` into the "Rule Name" field and presses the "New" button. The end user selects the communities and services to which this rule applies. For simplicity in this example, the end user wants to allow DNS from any internal nodes to any other internal nodes and therefore selects an Initiator community of hosts `Inside_Nodes`, a service of DNS, and a Target community of hosts

25 `Inside_Nodes`. The end user then presses the "Add Selected" button for each in turn to create a rule as shown in Fig. 10b, where Fig. 10b shows a dialog

box for generating a new rule according to the preferred embodiment of the invention.

Next the end user generates a new policy specification file and runs policy
5 monitor. The end user returns to the query tool and presses the "Network
Events" button again to get a new rule view dialog box. The topmost
"Execution Run" is now the output from the processing just completed. The
end user presses the "Query" button and can now see that DNS traffic from
10.5.63.143 to 10.5.63.6 is now conformant to the policy as shown in Fig. 10c,
10 where Fig. 10c shows the communities of the policy specification.

Detailed Description of Components

The preferred embodiment of the invention incorporates the following
components, detailed description of which follows below.

15

The Policy Generator Tool

The preferred embodiment of the invention provides a policy generator tool, or
simply policy generator, equally referred to as policy wizard, that provides a
level of abstraction on top of the policy language, and which simplifies the
20 process of creating an initial policy based on gross characteristics of a
network at the IP level, such as policy domains, communities of hosts,
servers, subnets, firewalls.

The policy generator provides a novel mechanism for translating desired
25 network security policy, such as corporate network security policy, into a

policy specification file that can be interpreted and implemented by a policy monitor mechanism.

Building a policy with the policy wizard involves: deciding on logical divisions
5 within the network, *i.e.* policy domains, grouping network nodes into logical communities, and expressing rules about which communities of hosts can provide what services to which communities of hosts.

High Level View of Policy Generation

10 The first step in building a basic policy is to define a high-level topology for the network. Not much detail is necessary. In the preferred embodiment of the invention, the network needs to be divided into bounded units called policy domains. In practice, the choice of a policy domain boundary is fairly obvious. Usually natural logical and physical boundaries in a network help define policy
15 domain boundaries. For example, firewalls and routers with packet filters commonly denote the important boundaries. When defining a simple policy, it is reasonable to ignore switches, bridges, hubs, and routers that connect interior subnets.

20 It is suggested that policy domains be as small as required by traffic monitoring limitations and as large as specification of rules allow. Rules are written about traffic visible in a policy domain. Traffic in a policy domain is logically considered to be visible anywhere within the policy domain even though networking elements, such as, for example, switches prevent such
25 visibility in most networks. By writing rules about traffic as though it is visible

anywhere within the policy domain, the same set of rules can be applied to network traffic anywhere within the policy domain.

5 It has been found that if a policy domain is too small, rules need to be duplicated for each extraneous policy domain. If a policy domain is too large, then the choice of a network traffic monitoring point can become overly constrained, or the ability to detect IP spoofing and rogue routers is lost.

Identify the Policy Domains

10 Fig. 11 shows a high-level view of an example network. An Intranet 1101 is connected to a DMZ 1102 through a firewall 1103. The DMZ 1102, in turn, connects through a router 1104 to the Internet 1105 and through a second router 1106 to an external corporate network 1107. In this example, an end user is only expected to be able to monitor traffic in the Intranet and DMZ, so
15 these two entities are declared to be policy domains. Rules in the policy only apply to allowed traffic in the DMZ and Intranet. The corporate network and Internet are viewed only as communities of hosts visible from within the policy domains.

20 It should be appreciated that the end user could choose to declare the Internet and Corporate network to be policy domains, but, by doing so, would only create unnecessary work because the end user does not intend to monitor traffic there. Any rules generated would thus never be used.

25 Add Perimeter Elements

In the preferred embodiment of the invention, the point of connection of a policy domain to the outside world is known as a perimeter element. For each perimeter element the set of nodes visible through it needs to be known and, for generating rules to detect IP spoofing and rogue routers, the MAC address
5 of the perimeter element itself needs to be known.

As an example, if an end user could sit inside a policy domain and look out through boundaries, it is probable that the end user would see a filtered
10 version of what is on the other side. Network address translation (NAT) can change the IP addresses seen through the boundary. For example, a proxying firewall may not let the end user see anything directly beyond a single IP address at the boundary. Filters may limit the view to only a few hosts when thousands are actually present.

15

Define Communities

In the preferred embodiment of the invention, communities consist of sets of IP addresses. They can be expressed as, for example, individual IP addresses, ranges of addresses, or subnet masks. Additionally, communities
20 can be composed of other communities. It is often the case that a community of nodes involves all nodes in some existing set except for a node or two. Communities are defined in terms of included elements and excluded elements.

25

Define Rules For Each Policy Domain

In the preferred embodiment of the invention, rules defined for a policy domain describe allowed transactions. For example, if no rules are written, the policy specifies that everything at the IP level or above is denied, although
5 this specification is not strictly true because typically auto-generated rules that apply to IP broadcast traffic and ICMP traffic within the policy domain exist. Rules create holes in this base layer that declares all traffic illegal.

Rules are defined in terms of initiator communities, target communities, and
10 the services allowed. Services consist of a set of port numbers and indicators of whether TCP or UDP protocols are used.

Using the Policy Generator

The preferred embodiment of the invention provides a front end for the policy
15 generator. It provides a user interface for entering and editing a simple policy. The front end reads and writes the current state of a policy from or to an intermediate file. The currently preferred extension for the intermediate file is .spw. When a policy has been specified to the satisfaction of the end user, it is written to an intermediate policy file for processing by the policy generator
20 backend that generates a formal policy specification file compatible with the policy monitoring system.

The front end allows the end user to edit policy domains, communities, services, and rules, to read and write the current policy from or to an
25 intermediate file, and to process the intermediate policy file into the formal policy specification file.

The preferred embodiment of the invention allows several instances of each editing process to be open simultaneously. The interaction is intended to feel very live. Data changed in one editing process should be reflected in the contents shown in other editing processes. For example, if a community is added in one community editing process, then it is immediately available for use in all editing processes. When building a policy, entities are first created, then filled in. From the time of creation they can be used throughout the policy. Consequently, a community or policy domain does not need to be fully specified in order to be used. However, to prevent errors in backend processing, all entities should be complete before the intermediate policy file is submitted to the backend for policy specification file generation.

In the preferred embodiment, only one policy is under development at any time. The front end starts up containing a default policy that is empty except for some predefined default services. This policy can be used as a starting point or an existing policy can be read from a saved intermediate policy file.

It has been found that it is best to use simple names in developing a policy and to use a name that makes sense from a predetermined point of reference, not a fully qualified name that makes sense from any point of reference. For example, it is better to give a rule a short, descriptive name such as, "Allow_Outgoing_Mail" than to give the rule a long name such as, "Allow_Mail_From_Intranet_To_Outside_Intranet".

For an in-depth understanding of the formal policy specification generated by the policy generator, or policy wizard, please refer to the section, Understanding the Wizard Generated Policy, below.

5

Collecting Packet Data

The preferred embodiment of the packet gathering component 128 is a program referred to as the harvester. It reads packets off the observed network 125 and writes them to either a packet capture file 126 or to a TCP socket that is connected to the policy monitor 100.

10

As an example, the harvester reads packets off the network when invoked as follows:

```
harvester -i eth0 -c 1000 -dump qs.dmp
```

15

In this example, 1000 packets are read from a network interface labeled 'eth0' and stored in file 'qs.dmp.'

20

The harvester can also be configured to read packet data and convert it to event data suitable for policy monitor 100. As an example, the harvester may be invoked as follows:

```
harvester -i eth0 -c 1000 -enc qs.dme
```

In this example, 1000 packets are read off the network interface labeled 'eth0', converted to event data suitable for policy monitor 100, and stored in the file 'qs.dme'.

- 5 The harvester can also be configured to read packet data, convert it to event data suitable for policy monitor 100, and stream such data directly to the policy monitor in real time. As an example, the harvester may be invoked as follows:

10 `harvester -i eth0 -c 1000 -enc 10.5.63.6:333`

In this example, 1000 packets are read off the network interface labeled 'eth0', converted to event data suitable for policy monitor 100, and transmitted in a TCP network stream to port 333 on the machine with IP address
15 10.5.63.6. This machine and TCP port may be configured so that the policy monitor 100 reads the data and processes it.

It should be appreciated that the events are transmitted as they are processed, so that the policy monitor 100 is able to see events shortly after
20 they occur on the observed network 125.

In this mode of operation, the policy monitor 100 is also able to pass information about policy dispositions back to the harvester. The harvester can use this information to make processing of packets more efficient. For
25 example, if the policy monitor 100 has determined that a given network event

is acceptable according to the policy, the monitor can sometimes expedite its protocol processing by skipping packets until the network event terminates.

Policy Monitor

- 5 The preferred embodiment of the invention provides a policy monitor component that provides a user interface, either graphical or command line, that allows the configuration of various options of the monitor, policy engine and logger.

10 **Monitor Configuration**

Monitor configuration allows the end user to configure the location of the input packet dump, policy to be used, and the specification of the monitoring point.

The ***Input dump file*** specifies the input file, in tcpdump format that is to be
15 used.

The ***Policy*** input specifies the .spm file that contains the policy specification to be used.

- 20 The ***Monitoring Point*** is a specification of where the Input dump file was collected. This name is derived from policy domain names that are specified in the policy wizard. For example, if a packet dump was collected in a policy domain named "Intranet" then the Monitoring Point name INTRANET_MONITOR should be used.

Monitor Logging Options

The monitor logging options allow the end user control of the location and the amount of data that gets written to the backend database.

- 5 The ***Execution Run Comment*** field allows the entry of freeform text that is added to the logs in the database to help identify this particular run of policy monitor.

- 10 ***ODBC Name*** provides the name of the ODBC source to which output data is written. ***The DB Username*** and ***DB password*** are the end user's database login information. The ***Save Password*** allows the program to save the password in the clear so that it does not need to be entered the next time the program is run.

15 Output Options

Output options allow the end user to specify whether the trace output from the monitor should be displayed in a console window (***Output to console***) or sent to a file (***Output to file:***).

20 Advanced Options

Advanced options allow more options to be set. In day to day operation, it is rare that such options need to be changed.

Advanced Monitor Configuration

- 25 An ***Assert DLL*** parameter allows specification of the name of the DLL to be used to verify condition and credential assertions. Note that if this DLL does

not match the version of the policy specified then this DLL is regenerated, overwriting the provided DLL.

A **Trace Options** parameter allows the end user to provide configuration of runtime trace options. This option affects the amount of output generated by the monitor. For a more efficient operation, this field should be left blank.

A **Certificate Dir** argument points to a directory that contains trusted CA root certificates in DER encoded form.

Advanced Packet Logging Options

The packet logging options section allows the configuration of the trace options to be provided by the low level packet monitor. The various logging options may be specified at a global level (by setting them for layer "-All-") or individually on a per-layer basis. Again it is to be noted that specifying logging options adversely affect the performance of the monitor.

The **Site Handle** parameter specifies a name that is associated with the particular company or site that is being monitored. It is used to segment a table that is used for IP-address name resolution within the output database.

Advanced Monitor Logging Options

The **Disable Logging** checkbox disables the writing of all logging data to the database. If logging is enabled then the remaining checkboxes provide for the enabling or disabling of the logging of network events with the given final disposition code. For example, if **Disable Logging** is not selected and only

Policy Error selected then the only network events that are logged to the database are those that resulted in a final disposition code of **POLICY_ERROR**.

- 5 During normal operation information about all protocol events within a network event is logged, even those that occurred after a final disposition was reached. An **Enable All Layer Logging** parameter can control this feature. When set on, all protocol events are logged to the database. When not set only those protocol events that are processed before a disposition is reached
- 10 are logged.

QueryTool

- The preferred embodiment of the invention provides a query tool to examine the data that was placed in the database. The preferred query tool allows the
- 15 following functions to be performed:

- Examining network events, such as protocol events, that are contained within the execution runs in the database;
- 20 ◦ Examining IP Connectivity for execution runs in the database;
- Editing and making user defined SQL queries to the database;
- Performing forward and reverse DNS lookups (using the current DNS
- 25 configuration);

- Viewing policy monitoring run information from the database, and selecting a default run for further viewing;
- Explicitly connecting to a specific database; and
- Turning on/off IP address to hostname resolution.

Other Tools

The preferred embodiment of the invention provides other tools discussed below.

Compiler

In its simplest form the compiler needs just a single argument that is the input policy specification file. This form is often all that is needed while doing initial development of a policy. It should be appreciated that the compiler is rarely used in standalone form since its function, with the exception of the -r flag, is subsumed into the policy monitor component.

Example Usage

During initial development a command such as the following could be used while getting rid of syntactic and semantic errors from the policy under development:

```
pmsCompiler.exe security.pms
```

Once compiler errors are gone, the end user is ready to generate pieces that are used to run the policy monitor. For example, the end user can use the command line:

5 `pmsCompiler.exe -d verify security.pms`

that compiles the security policy, and generates a verification DLL named "verify.dll".

10 **Compiler Options**

The following arguments in Table C may be provided to the example `pmsCompiler.exe`.

Table C

15 `pmsCompiler -? -r`

`-c <cxx-file> -d <dll-file>`

`<policy-file>*`

`-c <cxx-file>`

20 Generate Credential and Condition assertion verification code to the named file. The suffix ".cxx" is appended to the name that is provided. This option is *rarely used* to allow the end user to look at the actual code that is used to verify assertions.

`-d <dll-file>`

25 Generate a DLL containing the assertion verification code to the named file. The suffix ".dll" is appended to the name that is provided. If the `-d` flag is used without the `-c` flag then the source code is written to a temporary file. This option is *often used* to generate

the assertion verification DLL. The alternative is to allow the runtime Policy Monitor to generate the DLL for itself.

-r

- 5 Generate a pseudo-english description of the policy to stdout. The output of this command is a useful starting point for a policy report to a customer.

-?

Display a usage string.

10

<policy-file>

The *required* policy specification (".pms") file.

-b <db-name>

- 15 Store information about the compiled policy in the named database. db-name is the name of a user data source that has been configured within Control Panels->ODBC. This argument is *rarely used*. The alternative is to allow the runtime Policy Monitor to write the policy to the database if needed.

-o <output-file>

- 20 Redirect compiler messages to stdout to the named output file. *Rarely used*.

-t <trace-opts>

Enable debug tracing. For more specific details try providing the argument "-t ?". This option is *rarely used* because it only provides information to allow debugging of the compiler itself.

25

-v

Use VisualC++ to preprocess macros rather than the internal preprocessor. This overrides the -n option. This option is *rarely used*.

-g

- 30 Add debug trace code, *i.e.* printf statements, to the generated Credential and Condition verification code. The generated code is compiled with symbol information (the C compiler -g flag). This option is *rarely used*.

-n

Do not run a preprocessor. C preprocessor macros such as #define and #include may be included within a policy file. This option specifies that the pre-compiler should not be run prior to actually compiling. This option is *rarely used*.

-z

Output the dump output of the parsed policy. This output looks remarkably similar to the input file with the comments stripped and some component definitions reordered.

Network Monitor

The preferred embodiment provides a streams-based network monitor that can be run in a standalone mode independent of the policy monitor. In this way it can be used to provide a detailed, streams-based view of the network traffic, or a subset thereof. For example, run in standalone mode is desirable when a particular protocol is not supported natively by the policy monitor and an end user desires to see raw data to gain an understanding of what is going on.

It should be appreciated that a convenient way of accessing such functionality is through the query tool.

Example Usage

The following invocation of the network monitor:

```
mon -ev 2 -l ALL=all C:\spm\quickstart\qs.dmp
```

examines the qs.dmp file, producing extremely verbose output for event 2 only.

Table D provides a list of network monitor options according to the invention.

Table D

5	Monitor Options
	mon [-log LAYER[=[-]option1,[-]option2...]]*
	[-n npkt] [-skip pkt] [-until endpkt]
	[-ev eventID] [-untilev eventid] [-justev eventid]
	[-noclients] dump_file
10	-log
	Only process the first npkt packets from the input data.
	-n npkt
	Only process the first npkt packets from the input data.
	-skip pkt
	Skip pkt packets before beginning to process the input data.
15	-until endpkt
	Only process data through the packet number provided is reached
	-ev eventID
	Only process the data starting at the given eventID.
	-untilev eventid
20	Only process the data through eventid. Note that to find the end of eventid, events with ids greater than eventid may be processed.
	-justev eventid
	Only process the data for eventid. Note that to find the end of eventid, events with ids greater than eventid may be processed. This option is the equivalent of -ev eventid -untilev eventid.
25	-noclients
	Do not generate any output for higher level protocols such as HTTP, FTP, etc.
	dump_file
	The dump file, in tcpdump/windump format, that contains the input data.
30	

Understanding the Wizard Generated Policy

Using the Policy Generation Wizard, a user specifies a network security policy in terms of the network services provided by certain hosts to other hosts in the network. When such policy is processed, the wizard generates a formal and more detailed description of the network security policy using the policy language. The policy language specification may then be used to analyze network traffic using the policy monitor tool. The results of this analysis can be studied using the query tool. An exemplary policy language is taught in *A Declarative Language for Specifying a Security Policy*, patent application number 09/479,781 (1/7/2000), also included in section, A Declarative Language for Specifying a Security Policy herein below.

Understanding the output of the preferred query tool requires understanding how the preferred wizard translates the high-level view of security policy it presents to its users into a set of policy language objects such as rules, credentials and dispositions.

Understanding the policy generation process involves the following:

- Understanding the predefined rules, credentials and dispositions;
- Understanding the implicit rules and credentials; and
- Understanding the explicit rules and credentials.

Predefined Rules, Credentials and Dispositions

Every policy generated by the wizard includes a set of predefined default rules for handling protocol events that do not conform to the user-defined policy *i.e.* rules that deny access, as well as rules for handling common network events not covered by the user policy. These rules and their dispositions are shown

5 in Table E and Table F, and further discussed below.

Table E

Rule	Protocol – Action	Disposition
<i>Ip_Deny</i>	IP – all	<i>Ip_Access_Denied</i>
<i>Icmp_Deny</i>	ICMP – all	<i>Icmp_Access_Denied</i>
<i>Udp_Deny</i>	UDP – all	<i>Udp_Access_Denied</i>
<i>Tcp_Deny</i>	TCP – all	<i>Tcp_Access_Denied</i>
<i>Http_Deny</i>	HTTP – all	<i>Http_Access_Denied</i>
<i>Ftp_Deny</i>	FTP – all	<i>Ftp_Access_Denied</i>
<i>Ssl_Deny</i>	SSL – all	<i>Ssl_Access_Denied</i>
<i>Ssh_Deny</i>	SSH – all	<i>Ssh_Access_Denied</i>

10 Table F shows the default rules for all the protocols supported by the policy monitor. The policy engine selects these rules when no other rule can be found that is satisfied by the protocol event.

Table F

Rule	Protocol – Action	Disposition
<i>Ip_Deny_Pure_Ip</i>	IP – PROTOCOL_UNKNOWN	<i>Deny_Pure_Ip</i>
<i>Tcp_Missed_Connections</i>	TCP – MISSED_CONNECT	<i>Warn_Missed_Tcp_Connect</i>
<i>Ftp_Ignore_Data_Connectio</i>	FTP – DATA_OPEN	<i>ok</i>

<i>Rule</i>	<i>Protocol – Action</i>	<i>Disposition</i>
<i>ns</i>		

Table G below shows rules that cover protocol events not addressed by the wizard's user interface. These are well understood events that can be separated from those handled by the default rules. *Ip_Deny_Pure_Ip* is assigned to IP associations whose payload is not one of the three well-known IP-based protocols (ICMP, UDP and TCP). *Tcp_Missed_Connections* is assigned to network events where the establishment of the TCP connection was not witnessed by the policy monitor. *Ftp_Ignore_Data_Connections* is assigned to all FTP data connections which, from a security policy monitoring perspective, can be safely ignored. It is noted that the preferred policy wizard generates other rules to deal with common protocol events as discussed below.

Table G shows the predefined dispositions used by all the rules in the generated policy. Associated with each disposition are its disposition code and severity, which may be used in the query tool to filter network events.

Table G

<i>Disposition</i>	<i>Disposition Code</i>	<i>Disposition Severity</i>
<i>ok</i>	OK	None
<i>policy-error</i>	POLICY_ERROR	CRITICAL
<i>Ip_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Deny_Pure_Ip</i>	ACCESS_DENIED	HIGH
<i>Monitor_Broadcasts</i>	OK	MONITOR

<i>Disposition</i>	<i>Disposition Code</i>	<i>Disposition Severity</i>
<i>Icmp_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Monitor_Icmp</i>	OK	MONITOR
<i>Udp_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Tcp_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Warn_Missed_Tcp_Connect</i>	OK	WARNING
<i>Ftp_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Http_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Ssl_Access_Denied</i>	ACCESS_DENIED	HIGH
<i>Ssh_Access_Denied</i>	ACCESS_DENIED	HIGH

It should be noted that *ok* and *policy-error* are actually built-in dispositions in the policy language. If *policy-error* is encountered it indicates an error in the processing of either the policy or the network traffic data by the policy monitor.

5 The meaning of the other dispositions is explained later in this document in the context of the rules in which they are used.

Finally, the wizard includes a set of predefined credentials that are combined with dynamically generated credentials and used in implicitly generated rules:

10

__*Multicast_Addresses* - a set of commonly used IP multicast addresses;

__*Local_Broadcast_Address* - the IP address used for non-directed local broadcasts (255.255.255.255); and

15

Zero_Ip_Address – a zero-valued IP address (0.0.0.0), commonly used by BOOTP clients;

It is noted that the double underscore prefix in these credential names is used to ensure that there aren't any name conflicts with credentials generated to represent user-defined communities and services.

Explicit Rules and Credentials

Every community defined by the user results in a credential of the same name. Because the scope of a community name is that of the entire policy specification, the resulting credential names need not be massaged to ensure uniqueness.

Service names are also global in scope. Because services and communities share the same name space, every service defined in the policy results in a credential whose name is constructed by prefixing the user-supplied service name with the underscore character. Thus, for example, the *Smb* service is represented by a credential named *Smb*.

Rule names, on the other hand, are only unique within the scope of a policy domain. Furthermore, if a user-defined rule addresses a service that is both a UDP and a TCP service, the wizard generates two rules, one for the UDP protocol and another for the TCP protocol. Thus, a rule name is constructed by prefixing the user-supplied name with the protocol name (*Udp_* or *Tcp_*) and the policy domain name.

For example, if the user defines a rule titled *Smb_Services* within a policy domain named *Intranet*, the wizard generates two rules, *Udp_Intranet_Smb_Services* and *Tcp_Intranet_Smb_Services*, for the UDP and TCP protocols respectively.

5

User-defined rules may also result in the generation of additional credentials.

When defining a rule, the user provides the following information:

- Zero, one, or more initiator communities;

10

- Zero, one, or more services; and

- Zero, one, or more target communities.

15

If more than one initiator community are specified, the wizard generates a credential that combines these communities into a union. The credential name is constructed by appending the word *_Initiator* to the user-supplied rule name, prefixed by the policy domain name. Using the example above, the wizard would create a credential named *Intranet_Smb_Services_Initiator*.

20

Likewise, if more than one target communities are specified, the wizard creates a credential representing their union and names it by appending the word *_Target* to the policy domain and rule names, e.g. *Intranet_Smb_Services_Target*).

25

However, if one or more services are specified they are combined with the target credentials according to the service type. For example, the *Smb* service (for the SMB protocol suite) and its like-named credential include ports that are used for both TCP and UDP. Thus, for the *Smb_Services* rule used

5 above, the wizard would generate the following additional credentials:

Udp_Intranet_Smb_Services_Target a n d
Tcp_Intranet_Smb_Services_Target. These credentials combine
Intranet_Smb_Services_Target (or a single target community) with the *_Smb*
credential and constitute the actual target credentials used in

10 *Udp_Intranet_Smb_Services* and *Tcp_Intranet_Smb_Services* respectively. It should be noted that, in many cases, the set of UDP and TCP services referenced in a rule have little, if any overlap.

If the end user does not specify any services the wizard uses the

15 *Intranet_Smb_Services_Target* credential (or a single target community credential) to identify the target principal.

Implicit Rules and Credentials

For each policy domain within the policy specification, the wizard

20 automatically generates a set of rules and credentials that define the valid IP-level traffic seen at the monitoring point within the domain. In addition, an ICMP rule is generated that handles all intradomain ICMP traffic, as well as a credential for the monitoring point in that domain.

25 The monitoring point credential is based on an agent descriptor string manufactured by the wizard. The agent descriptor is constructed by

converting the policy domain name to uppercase and appending to it the word *_MONITOR*. Thus, for example, a policy domain named *Intranet* is assigned the agent descriptor:

5 *INTRANET_MONITOR.*

Note that this is the agent descriptor to be used in the policy monitor when analyzing data collected at this monitoring point.

10 The monitoring point credential itself is named by appending the word *_Monitors* to the policy domain's name. In the example above, the credential is named *Intranet_Monitors*.

The wizard segregates all intradomain ICMP traffic (common on an enterprise
15 network) by use of a rule that assigns it the disposition *Monitor_Icmp*. The rule is named by combining the protocol name with the domain name using the word *_Within*. For example, in the *Intranet* policy domain the rule is named *Icmp_Within_Intranet*.

20 IP traffic is described by a set of rules that systematically enumerate all valid IP-level traffic within the policy domain, between hosts in the policy domain and external hosts, and between external hosts through the policy domain (when more than one perimeter element is present). Most of these rules provisionally allow IP traffic, letting the subsequent protocol layers (ICMP,
25 UDP, TCP, etc.) determine if the traffic is indeed allowed either by a user-defined (explicit) rule or by a predefined rule.

The first IP rule provisionally allows all intradomain IP traffic. It is named by combining the protocol name with the domain name using the word *_Within* (e.g., *Ip_Within_Intranet*). In the absence of a higher-level protocol within an intradomain IP association, the rule assigns the network event a disposition of *Deny_Pure_Ip*, i.e. its final outcome.

The intradomain IP rule uses the policy domain's defining community as its target principal. However, it generates another credential to be used as the initiator. This credential combines the defining community with the predefined credential for zero-valued IP addresses (*__Zero_Ip_Address*). The generated credential is named by appending the word *_Initiator* to the generated rule name, e.g. *Ip_Within_Intranet_Initiator*.

Another intradomain IP rule is used to segregate typical broadcast and multicast traffic within an enterprise network. It is named by combining the protocol name with the domain name using the words *_Broadcasts_Within*, e.g. *Ip_Broadcasts_Within_Intranet*. Its initiator principal is the same as that used for the general intradomain traffic, e.g. *Ip_Within_Intranet_Initiator*. Its target is a new credential constructed by combining the predefined credentials *__Multicast_Addresses* and *__Local_Broadcast_Address* with the directed broadcast addresses for all the subnets within the policy domain's defining community. The new credential is named by appending the word *_Target* to the rule name e.g. *Ip_Broadcasts_Within_Intranet_Target*.

The intradomain broadcast and multicast traffic is assigned the disposition *Monitor_Broadcasts*.

Traffic between hosts in the policy domain and external hosts is described by
5 a set of rules whose complexity depends on how much information the user
supplied about the topology of the network. Specifically, it depends on how
many perimeter elements were specified and on whether or not the interface
addresses, *i.e.* MAC addresses, of the perimeter elements are included in the
policy specification.

10

If there are external communities associated with at least one perimeter
element for which the interface address is not known, the wizard generates a
credential combining all such communities in a single union unless there is
only one such community, in which case its credential already exists. This
15 credential is named by combining the policy domain name with the string
_External_Communities, e.g. *Intranet_External_Communities*.

The wizard then generates two rules defining the traffic between hosts internal
to the policy domain and these external communities. The wizard names
20 these rules by combining the protocol name with the domain name and the
string *_To_External_Communities* or *_External_Communities_To*, depending
on the direction of the IP traffic, e.g. *Ip_Intranet_To_External_Communities*
for outbound traffic and *Ip_External_Communities_To_Intranet* for inbound
traffic.

25

The credentials used alternately as the initiator and target principals for these rules are the policy domain's defining community and the aforementioned credential for the external communities. The rules provisionally allow the IP traffic to flow, subject to other rules for higher level protocols. In the absence
5 of a higher-level protocol within the network event, the rule assigns it a disposition of *Deny_Pure_Ip*, i.e. its final outcome.

External communities visible through one or more perimeter elements whose interface addresses are known, are handled by a separate set of rules, two
10 per perimeter element. For each perimeter element, the wizard starts by creating a credential that combines one or more credentials for one or more external communities visible through it with the perimeter element's interface address. Such credential is named by combining the domain name with the perimeter element name and the string *_Communities*. For example, external
15 communities visible through a perimeter element named *Firewall* are described by a credential named *Intranet_Firewall_Communities*.

The wizard then generates two rules defining the traffic between hosts internal to the policy domain and the external communities visible through this
20 perimeter element. The wizard names these rules by combining the protocol name, the domain name, the perimeter element name and the word *_To*, e.g. *Ip_Intranet_To_Intranet_Firewall* for outbound traffic and *Ip_Intranet_Firewall_To_Intranet* for inbound traffic.

25 The credentials used alternately as the initiator and target principals for these rules are the policy domain's defining community and the aforementioned

credential for the external communities. The rules provisionally allow the IP traffic to flow, subject to other rules for higher level protocols. In the absence of a higher-level protocol within the network event, the rule assigns it a disposition of *Deny_Pure_Ip*, i.e. its final outcome.

5

Finally, if there is more than one perimeter element associated with the policy domain, the wizard generates rule-pairs that describe the traffic between external communities visible through specific perimeter elements as well as external communities visible through any perimeter element, i.e. those without associated interface addresses. The rules are named by combining the names of each pair of perimeter elements with the protocol name, the policy domain name and with the word *_To*, in the case of addressable perimeter elements, or with the string *_External_Communities*, for all other external communities. An additional rule is generated to cover traffic between external communities not associated with an addressable perimeter element and is named by combining the protocol name with the domain name and the string *_Between_External_Communities*.

Thus, if the *Intranet* domain used as an example in this section were to have a second (addressable) perimeter element named *Router* and a third non-addressable perimeter element (whose name is unimportant), the wizard would generate the following rules to cover all traffic amongst their respective external communities:

25 ◦ *Ip_Intranet_Firewall_To_Intranet_Router*

- *Ip_Intranet_Router_To_Intranet_Firewall*
- *Ip_Intranet_Firewall_To_External_Communities*
- 5 ◦ *Ip_External_Communities_To_Intranet_Firewall*
- *Ip_Intranet_Router_To_External_Communities*
- *Ip_External_Communities_To_Intranet_Router*
- 10 ◦ *Ip_Intranet_Between_External_Communities*

Table H and Table I summarize all the implicit rules and credentials generated for the example policy domain *Intranet*. The policy domain includes two
 15 perimeter elements with a specified interface address (*Firewall* and *Router*) and a third non-addressable perimeter element.

Table H

<i>Credential</i>	<i>Comment</i>
<i>Intranet_Monitors</i>	Uses agent descriptor INTRANET_MONITOR
<i>Ip_Within_Intranet_Initiator</i>	Defining community plus zero-valued IP address
<i>Ip_Broadcasts_Within_Intranet _Target</i>	Combines standard multicast addresses with local broadcast and directed broadcast addresses
<i>Intranet_External_Communities</i>	Combines all external communities not associated with addressable perimeter elements
<i>Intranet_Firewall_Communities</i>	Combines all external communities visible through the <i>Firewall</i> perimeter element

Credential	Comment
Intranet_Router_Communities	Combines all external communities visible through the Router perimeter element

Table I

Rule	Credentials (I – Initiator T – Target)	Disposition (I – Immediate F – Final)
Ip_Within_Intranet	I: <i>Ip_Within_Intranet_Initiator</i> T: <i>Intranet</i>	I: continue F: <i>Deny_Pure_Ip</i>
Ip_Broadcasts_Within_Intranet	I: <i>Ip_Within_Intranet_Initiator</i> T:	I: <i>Monitor_Broadcast</i>
	<i>Ip_Broadcasts_Within_Intranet_Target</i>	
Icmp_Within_Intranet	I: none (ignore) T: none (ignore) Note: uses <i>Ip_Within_Intranet</i> as prerequisite	I: <i>Monitor_Icmp</i>
Ip_Intranet_To_External_Communities	I: <i>Intranet</i> T: <i>Intranet_External_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
Ip_External_Communities_To_Intranet	I: <i>Intranet_External_Communities</i> T: <i>Intranet</i>	I: continue F: <i>Deny_Pure_Ip</i>
Ip_Intranet_To_Intranet_Firewall	I: <i>Intranet</i> T: <i>Intranet_Firewall_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
Ip_Intranet_Firewall_To_Intranet	I: <i>Intranet_Firewall_Communities</i> T: <i>Intranet</i>	I: continue F: <i>Deny_Pure_Ip</i>
Ip_Intranet_To_Intranet_Router	I: <i>Intranet</i> T: <i>Intranet_Router_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>

Rule	Credentials (I – Initiator T – Target)	Disposition (I – Immediate F – Final)
<i>Ip_Intranet_RouterTo_Intranet</i>	I: <i>Intranet_Router_Communities</i> T: <i>Intranet</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_Intranet_Firewall_To_Intranet_Router</i>	I: <i>Intranet_Firewall_Communities</i> T: <i>Intranet_Router_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_Intranet_Router_To_Intranet_Firewall</i>	I: <i>Intranet_Router_Communities</i> T: <i>Intranet_Firewall_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_Intranet_Firewall_To_External_Communities</i>	I: <i>Intranet_Firewall_Communities</i> T: <i>Intranet_External_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_External_Communities_To_Intranet_Firewall</i>	I: <i>Intranet_External_Communities</i> T: <i>Intranet_Firewall_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_Intranet_Router_To_External_Communities</i>	I: <i>Intranet_Router_Communities</i> T: <i>Intranet_External_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_External_Communities_To_Intranet_Router</i>	I: <i>Intranet_External_Communities</i> T: <i>Intranet_Router_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>
<i>Ip_Intranet_Between_External_Communities</i>	I: <i>Intranet_External_Communities</i> T: <i>Intranet_External_Communities</i>	I: continue F: <i>Deny_Pure_Ip</i>

Logging and Reporting Modules

The preferred embodiment of the invention provides logging and reporting modules, as described herein with reference to Fig. 1a. As the policy engine
5 module 102 reaches dispositions on network events, it passes the network event object to the logging module 103.

The preferred embodiment of the invention also provides an alarm script 155. As the policy engine module 102 reaches dispositions on network events of a

certain disposition severity, for example, CRITICAL or HIGH, the alarm script is invoked to provide expedited alerting of the disposition.

The following algorithm is used to enter the data into the database 104.

5

- During initialization of the logging module 103, the database 104 is tested to see if it contains a policy that matches the MD5 hash of the policy 105 currently being used by the policy engine 102. If no such policy is found then the policy details are added to the database 104;

10

- with each network event passed to the logging module 103, if logging of network events is enabled, then:

15

- if the final disposition of the network event matches one of the list of dispositions that is to be logged, then:

- add the network event to the buffer of network events, flushing the buffer to the database 104 if it is full;

20

- loop through each of the protocol events contained in the network event;

25

- if the initiator and responder principals have not been already added to the database 104 then do so, caching the database keys for later use; and

- add the protocol event to the buffer of network events,
flushing the buffer to the database 104 if it is full.

On a periodic basis report statistics 161 are sent across a secure channel to a
5 secure, customer accessible server 162. The preferred embodiment of the
invention uses the following algorithm.

- A report script 160 described is used to generate a report 161 for the
configured or predetermined time period. An example of a list of preferred
10 acquired or calculated statistics or intermediate steps is contained in Table
J below;
- The report 161 is then packaged using the tar command and PGP to
encrypt the resulting file using the public key of a recipient email account;
15 and
- This encrypted file is then emailed to the recipient email account.

It should be appreciated that an equally preferred embodiment performs name
20 resolution on packet data after the packet data has been collected, rather than
concurrent with collecting the packet data. An advantage to such name
resolution technique is that name resolution after collection is removed from
real-time processing, thereby rendering name resolution more efficient.

25 On the receiving secure server 162 the following algorithm is invoked on the
received email message.

- PGP is used to decrypt the received encrypted tar file;
 - Tar is used to extract the report data;
- 5
- The report data is then processed to link the report into the reporting website 164 for the client; and
 - Any supplied protocol event data is then stored in a reporting database
- 10 165.

Upon accessing the reporting website 164 the client is able to peruse the reports that have been generated, access the protocol event data stored in the database 165 via a cgi script.

15

Table J

20

- Generate network events in subsidiary web files, based on execution run;
 - Generate network events table,
 - Generate table for URL's and status codes;
 - Find events of interest;
 - Check for all execution runs being in sequence;
 - Give best optimization for queries;
 - Compute number of events and number of exceptions;
 - Apply definitions of log severity and disposition code in order of criticality;
- 25
- Apply query to several execution runs at a time, collect results;
 - Select key disposition and key policy rule first, to be able to find distinct disposition and policy rule;
 - Determine sort order for disposition and policy rule table; and

- Generate a list of dispositions in the selected events, counting how many events were generated by each.

Automated Generation of an English Language Representation of a Formal

5 Network Security Policy Specification

The preferred embodiment of the invention uses a formal specification of network security policy that is to be enforced on a network. This specification provides a precise, compact description of network security policy. However, it is difficult for a layperson to understand. In order to allow comprehension of the policy by non-technical staff within a user's organization the parser module (Fig. 1 150) is used to generate an English language description of the policy. This description is simple enough to be understood, yet captures the salient details of the policy. It will be appreciated that the invention generated a representation in a human readable language, such as english, those skilled in the art will recognize that the invention may generate representations in any human readable language.

The preferred embodiment of the invention provides the following algorithm for generating the English language representation. The algorithm comprises the following:

- Loading the policy into the parser from its text representation; and
- Looping through all supported protocols, from the highest level protocols to the lowest;

- Sorting the rules for this protocol into ranked order; and
- Looping through these rules from the highest ranking to the lowest;

- 5 ◦ Generating a text description of the rule using the algorithm below.
 If an HTML flag has been set then format the text into a HTML
 table; and
- 10 ◦ Append this description to a collection of descriptions already
 generated.

The preferred embodiment of the invention provides the following rule algorithm to generate an English language representation of a single policy language rule. The algorithm is described with reference to Fig. 12. The

15 algorithm outputs the name of the rule at hand (2001). It then proceeds to output the agent's name (2002), where the agent is the subject network monitor(s) to which the policy applies. The algorithm then loops through all protocol and action combinations (2003). If the action is to be ignored (2004), then the rule applies to the whole protocol (2005). Otherwise, the rule applies

20 to certain actions only (2014). The algorithm then looks at the immediate outcome for the rule (2006). The algorithm then outputs the corresponding directive for the outcome (2007). If any conditions exist on the disposition, then the algorithm outputs the conditions (2008). The algorithm looks at the final outcome (2011), then outputs the corresponding final outcome of the rule

25 (2012). If any conditions exist on the disposition, then the algorithm outputs the conditions (2013). If the rule applies to a particular initiator or target, then

the algorithm outputs the initiator or target name (2009). Otherwise, the algorithm outputs a general inclusive name, such as, for example, "anyone." The algorithm then checks for prerequisites (2010). If any are discovered, the algorithm then outputs such prerequisites.

5

For an example of the rule algorithm discussed above, Table K below shows code to the example implementation.

Table K

```

10      if (isBuiltin())
           return;

           Bool processedImmediate = false;
           Bool immediateDefaultContinue = false;
15      Bool capitalize = true;
           string str;
           string protocol;

           // output the table row start
           if (html) str = "\n<tr><p>"; else str = "\n\n";

           // output the rule name
           if (html)
25      str += "<TD WIDTH=\"10%\" VALIGN=\"TOP\"><B>" + getName() + "<a name = \"" + getName() +
           "\"></a></B></TD>";
           else
           str += "Rule " + getName() + ": ";

           // output the agent name
           string agentName;
30      if (getAgent() == 0)
           agentName = "All Monitors";
           else
           agentName = getAgent()->getName();
35      if (html)
           str += "<TD WIDTH=\"5%\" VALIGN=\"TOP\">" + agentName + "</TD>";

           // start the cell for the description
           if (html)
40      str += "<TD WIDTH=\"85%\" VALIGN=\"TOP\">";

           // loop through the protocol and action combinations
           Bool first = true;
           for (PrsUnion::const_iterator t0 = _protocol->begin();
45      t0 != _protocol->end();
           t0++)

```

```

{
    for (PrsUnion::const_iterator t2 = _action->begin();
        t2 != _action->end();
        t2++)
    {
        if (first)
            first = false;
        else
            protocol += ", ";

        // if the action is ignore then it applies to the whole protocol
        if ((*t2)->getStringRepresentation() != PrsConst::META_IGNORE)
            protocol += (*t0)->getStringRepresentation() + "-" + (*t2)->getStringRepresentation() + " ";
        else
            protocol += (*t0)->getStringRepresentation() + " ";
    }
}

// look at the outcome to figure what we do with this traffic

// is there an immediate clause
if (_immediate != 0)
{
    // output text based on the code
    string code = _immediate->getDefault()->getCode();
    if (code == PrsConst::DISPCODE_OK)
    {
        capitalize ? str += "Allow " : str += "allow ";
        capitalize = false;
    }
    else if (code == PrsConst::DISPCODE_CONTINUE)
    {
        if (_final->getDefault()->getCode() == PrsConst::DISPCODE_OK)
            capitalize ? str += "Provisionally allow " : str += "provisionally allow ";
        else if (_final->getDefault()->getCode() == "POLICY_ERROR")
            ; // say nothing... this is the default
        else
            capitalize ? str += "Provisionally deny " : str += "provisionally deny ";
    }

    immediateDefaultContinue = true;
}
else
{
    capitalize ? str += "Deny " : str += "deny ";
    capitalize = false;
}
str += protocol;

if ((_immediate->getGuards()) != 0 && (_immediate->getGuards()->size() != 0)) /* KGS &&
immediateDefaultContinue */
{
    if (_immediate->getGuards()->size() == 1)
        str += "with condition (";
    else
        str += "with conditions (";
    first = true;
    for (std::vector<PrsGuardedDisposition*>::const_iterator cond = _immediate->getGuards()-
>begin();

```

```

cond != _Immediate->getGuards()->end();
cond++)
{
    if (first)
        first = false;
    else
        str += ", ";
    if (html) str += "<|>";
    str += (*cond->getGuard()->getName());
    if (html) str += "</|>";
}
str += ", ";
}
processedImmediate = true;
}

// is there a final clause
if (_final != 0)
{
    if (!processedImmediate)
    {
        // output text based on the code
        string code = _final->getDefault()->getCode();
        if (code == PrsConst::DISPCODE_OK)
        {
            capitalize ? str += "Provisionally allow " : str += "provisionally allow ";
            capitalize = false;
        }
        else if (code == "POLICY_ERROR")
            ; // say nothing... this is the default
        else
        {
            capitalize ? str += "Provisionally deny " : str += "provisionally deny ";
            capitalize = false;
        }
        str += protocol;

        if ((_final->getGuards()) != 0 && (_final->getGuards()->size() != 0))
        {
            if (_final->getGuards()->size() == 1)
                str += "with condition (";
            else
                str += "with conditions (";
            Bool first = true;
            for (std::vector<PrsGuardedDisposition*>::const_iterator cond = _Immediate->getGuards()-
>begin();
                cond != _Immediate->getGuards()->end();
                cond++)
            {
                if (first)
                    first = false;
                else
                    str += ", ";
                if (html) str += "<|>";
                str += (*cond->getGuard()->getName());
                if (html) str += "</|>";
            }
            str += ", ";

```

```

    }
}
else
{
5      // output text based on the code
      string code = _final->getDefault()->getCode();
      if (!immediateDefaultContinue)
      {
10         if (code == PrsConst::DISPCODE_OK)
             str += "but provisionally allow ";
            else if (code == "POLICY_ERROR")
                ; // say nothing... this is the default
            else
                str += "but provisionally deny ";
15     }
    if ((_final->getGuards()) != 0 && (_final->getGuards()->size() != 0))
    {
        str += "with conditions (";
        Bool first = true;
20     for (std::vector<PrsGuardedDisposition*>::const_iterator cond = _immediate->getGuards()-
        >begin();
            cond != _immediate->getGuards()->end();
            cond++)
        {
25             if (first)
                first = false;
            else
                str += ", ";
            if (html) str += "<|>";
            str += (*cond)->getGuard()->getName();
            if (html) str += "</|>";
30         }
        str += "), ";
    }
35 }
}

if (html)
    str += "from <|>"
40     + (_initiator->getCredential() ? _initiator->getCredential()->getName() : "anyone")
    + "</|> to <|>"
    + (_target->getCredential() ? _target->getCredential()->getName() : "anyone")
    + "</|>";
else
45     str += "from "
    + (_initiator->getCredential() ? _initiator->getCredential()->getName() : "anyone")
    + " to "
    + (_target->getCredential() ? _target->getCredential()->getName() : "anyone");

50 if (getPrerequisite() != 0)
{
    str += ", provided that ";
    Bool first = true;
55     for (vector<const PrsRule*>::const_iterator t3 = _prerequisite->begin();
        t3 != _prerequisite->end();
        t3++)
    {
        if (first)

```

```

        first = false;
    else
        str += " or ";
    if (html)
        str += "<|><a href=\"#" + (*t3)->getName() + "\">" + (*t3)->getName() + "</a></|>";
    else
        str += (*t3)->getName();
    }
    str += " is true.";
}

// start the cell for the description
if (html)
    str += "</TD></TR>";
else
    str += " (Agent " + agentName + ").";

ostm << str.c_str();

```

For an example of an output file generated by the main algorithm discussed above, Table L shows the example of the output in table format. For an example of a policy specification file that can be used as input into the main algorithm discussed above, refer to Table P below.

Table L

Rules for protocol HTTP

Http_Blocked_Service_Violation	All Monitors	Deny HTTP from anyone to anyone, provided that <i>Tcp Blocked Services</i> is true.
Http_Deny	All Monitors	Deny HTTP from <i>anyone</i> to <i>anyone</i>

Rules for protocol FTP

Ftp_Blocked_Service_Violation	All Monitors	Deny FTP from anyone to anyone, provided that <i>Tcp Blocked Services</i> is true.
--------------------------------------	-----------------	--

Ftp_Deny	All Monitors	Deny FTP from <i>anyone</i> to <i>anyone</i>
Ftp_Anonymous_Authentication	All Monitors	Allow FTP-CONTROL_AUTHENTICATE with condition (<i>Authentication_Rejected</i>), from <i>Anon_User</i> to <i>anyone</i>
Ftp_Validate_Password	All Monitors	Allow FTP-CONTROL_AUTHENTICATE with conditions (<i>Authentication_Rejected</i> , <i>Strong_Password</i>), from <i>anyone</i> to <i>anyone</i>
Ftp_Ignore_Data_Connections	All Monitors	Allow FTP-DATA_OPEN from <i>anyone</i> to <i>anyone</i>

Rules for protocol SSH

Ssh_Validate_Handshake	All Monitors	Allow SSH-HANDSHAKE , SSH-SESSION_ABORTED with conditions (<i>Ssh_Authentication_Failed</i> , <i>Ssh_Authentication_Aborted</i> , <i>Ssh_Secure_Authentication_Modes</i>), from <i>anyone</i> to <i>anyone</i>
Ssh_Blocked_Service_Violation	All Monitors	Deny SSH from <i>anyone</i> to <i>anyone</i> , provided that <i>Tcp_Blocked_Services</i> is true.
Ssh_Deny	All Monitors	Deny SSH from <i>anyone</i> to <i>anyone</i>

Rules for protocol SSL

Ssl_Validate_Handshake	All Monitors	Allow SSL-HANDSHAKE with conditions (<i>Authentication_Rejected</i> , <i>Ssl_Session_Qos</i>), from <i>anyone</i> to <i>anyone</i>
Ssl_Blocked_Service_Violation	All Monitors	Deny SSL from <i>anyone</i> to <i>anyone</i> , provided that <i>Tcp_Blocked_Services</i> is true.

Ssl_Deny	All Monitors	Deny SSL from <i>anyone</i> to <i>anyone</i>
Ssl_Missed_Handshakes	All Monitors	Allow SSL-MISSED_HANDSHAKE from <i>anyone</i> to <i>anyone</i>

Rules for protocol TCP

Tcp_Blocked_Services_Response	All Monitors	Deny TCP-ABORT , TCP-CLOSE , TCP-TIMEOUT with condition (Tcp_Data_Xfer), from anyone to anyone, provided that <u>Tcp_Blocked_Services</u> is true.
Tcp_Connection_Terminated	All Monitors	Allow TCP-ABORT , TCP-CLOSE , TCP-TIMEOUT from <i>anyone</i> to <i>anyone</i>
Tcp_Deny	All Monitors	Provisionally deny TCP from <i>anyone</i> to <i>anyone</i>
Tcp_X_Shsh_From_Clouds_To_Cgi_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from C l o u d s t o Tcp_X_Shsh_From_Clouds_To_Cgi_Provisional_Target
Tcp_X_Spm_Colloc_Traffic	X_Monitors	Allow TCP-CONNECT from Modin to Tcp_X_Spm_Colloc_Traffic_Target
Tcp_X_Spm_Colloc_Traffic_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from M o d i n t o Tcp_X_Spm_Colloc_Traffic_Provisional_Target
Tcp_X_Ssh_From_Monkey_To_Fluffy_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from M o n k e y t o Tcp_X_Ssh_From_Monkey_To_Fluffy_Provisional_Target
Tcp_X_X_Loghost_Traffic	X_Monitors	Allow TCP-CONNECT from X _ W e b _ S e r v e r s t o

		Tcp_X_X_Loghost_Traffic_Target
Tcp_X_Dns_From_Colloc_To_Dns_Server	X_Monitors	Allow TCP-CONNECT from X_Colloc_Subnet to Tcp_X_Dns_From_Colloc_To_Dns_Server_Target
Tcp_X_Port_1984_Traffic	X_Monitors	Allow TCP-CONNECT from X_Colloc_Subnet to Tcp_X_Port_1984_Traffic_Target
Tcp_X_Ssh_To_Web_Server	X_Monitors	Allow TCP-CONNECT from X_Ssh_To_Web_Server_Initiator to Tcp_X_Ssh_To_Web_Server_Target
Tcp_X_Ssh_From_Fluffy_To_Monkey_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from Fluffy to Tcp_X_Ssh_From_Fluffy_To_Monkey_Provisional_Target
Tcp_X_Ssh_From_X_To_X_Web_Servers_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from X_Ssh_From_X_To_X_Web_Servers_Provisional_Initiator to Tcp_X_Ssh_From_X_To_X_Web_Servers_Provisional_Target
Tcp_X_Http_From_Any_To_All_Web_Servers_Provisional	X_Monitors	Provisionally allow TCP-CONNECT from anyone to Tcp_X_Http_From_Any_To_All_Web_Servers_Provisional_Target
Tcp_X_Stmp_From_All_To_X	X_Monitors	Allow TCP-CONNECT from X_Stmp_From_All_To_X_Initiator to _Stmp
Tcp_Blocked_Services	All Monitors	Provisionally deny TCP-CONNECT from anyone to anyone

Tcp_Missed_Connections	All Monitors	Allow TCP-MISSED_CONNECT from <i>anyone to anyone</i>
Tcp_Blocked_Services_Violation	All Monitors	Deny TCP-PROTOCOL_UNKNOWN from anyone to anyone, provided that <u>Tcp_Blocked_Services</u> is true.
Tcp_Unknown_Protocol	All Monitors	Deny TCP-PROTOCOL_UNKNOWN from <i>anyone to anyone</i>

Rules for protocol UDP

Udp_X_Dns_From_Colloc_To_Dns_Server	X_Monitors	Allow UDP-ASSOCIATION from <u>X_Colloc_Subnet</u> to <u>Udp_X_Dns_From_Colloc_To_Dns_Server_Target</u>
Udp_Deny	All Monitors	Deny UDP from <i>anyone to anyone</i>

Rules for protocol ICMP

Icmp_Within_X	X_Monitors	Allow ICMP-ASSOCIATION from anyone to anyone, provided that <u>Ip_Within_X</u> is true.
Icmp_Deny	All Monitors	Deny ICMP from <i>anyone to anyone</i>

Rules for protocol IP

Ip_Directed_Broadcasts_Within_X	X_Monitors	Allow IP-ASSOCIATION from <u>Ip_Within_X_Initiator</u> to <u>Ip_Directed_Broadcasts_Within_X_Target</u>
Ip_External_Communities_To_X	X_Monitors	Provisionally deny IP-ASSOCIATION from <u>X_External_Communities</u> to <u>X_Colloc_Subnet</u>
Ip_X_To_External_Communities	X_Monitors	Provisionally deny IP-ASSOCIATION from <u>X_Colloc_Subnet</u> to <u>X_External_Communities</u>

Ip_Within_X	X_Monitors	Provisionally deny IP-ASSOCIATION from <i>Ip_Within_X_Initiator</i> to <i>X_Coloc_Subnet</i>
Ip_Non_Directed_Broadcasts_Within_X	X_Monitors	Allow IP-ASSOCIATION from <i>Ip_Within_X_Initiator</i> to __Generic_Multicast_And_Broadcast_Addresses
Ip_Deny	All Monitors	Deny IP from <i>anyone</i> to <i>anyone</i>
Ip_Unknown_Protocol	All Monitors	Deny IP-PROTOCOL_UNKNOWN from <i>anyone</i> to <i>anyone</i>

Algorithm for Efficient Rule Evaluation

The preferred embodiment of the invention comprises a technique for a policy engine internally to organize policy rules in order to effect an efficient evaluation of protocol events at runtime. Evaluation of a protocol event entails selecting one or more applicable policy rules using an evaluation algorithm. The preferred evaluation algorithm is described in *A Declarative Language for Specifying a Security Policy*, U.S. patent application number 09/479,781 (1/7/2000). An excerpt describing the preferred evaluation algorithm is provided below in Table P.

Using this technique, policy rules are organized in a manner that minimizes the number of rules that need to be considered when determining the set of rules applicable to a given protocol event. The algorithm is described with reference to Fig. 13 as follows:

- Create a first associative array, such as, for example, agent-to-protocols, where the key is an agent descriptor and the value is a reference to a second associative array with all the policy rules applicable to network traffic monitored by that agent (3001);

5

- Create a second associative array, such as, for example, protocol-to-actions, where the key is a protocol name and the value is a reference to a third associative array with all the policy actions applicable to that protocol (3002).

10

- Create a third associative array, such as, for example, action-to-rules, where the key is a protocol action and the value is a reference to the policy rules applicable to that protocol action (3003). The rules referenced in this list (3004) are sorted in decreasing order of rank number, taking into account any constraints such as, for example, rank-above, that might be present. Rules with the same rank number are ordered in the lexical order of their names.

15

It should be noted that the same rule can be referenced by different lists of ordered rules and, in each list, can have different rank numbers because the ranking of a rule is relative to the ranking of the other rules in the same list.

20

Assessment Tool

The preferred embodiment of the invention provides an assessment tool that allows the discussed technique for continuously assessing the security of a system to be applicable to both long-term and short-term network

25

assessment. The tool provides an additional dimension to network assessment. That is, it provides the ability to capture and classify large volumes of network traffic efficiently, based on a formal policy which describes permitted traffic. The tool adds network usage to the known list of features
5 discussed in an assessment framework.

It has been found through field experience that the invention can be useful in the following contexts:

- 10 ◦ Identifying services that were not mentioned by the system administration staff of a network that is being assessed;
- Identifying usage patterns of critical machines. In an assessment framework, this applies to typical usage patterns, because a long-term
15 deployment of the invention is needed to continuously analyze and monitor changes in usage or rare aberrant behavior;
- Identifying services; and
- 20 ◦ Analyze routing patterns. It should be appreciated that subnets are not scanned.

It should be appreciated that using the invention as a supplemental process in performing network assessments results in at least the following benefits:

25

- Rather than providing an inference of possible network behavior that is based on what hosts are configured to do, the network behavior is directly analyzed based on direct observation of data traffic;

- 5
- Rather than basing security analysis on a static snap-shot of the network environment as it existed at a particular moment, the analysis is based on a dynamic recording of network behavior over some non-trivial amount of time. As an analogy, traditional known network vulnerability scans take still photographs, while the invention takes a motion picture;

10

- Instead of relying on the accuracy of information provided by the customer point of contact through an interview process, the invention provides specific and tangible data points for discussion that facilitates the interview process and educates the customer on problems in an immediate
- 15 feedback loop; and

15

- Because the invention is policy based, and because of the rigor built into the policy language and analysis engine, the otherwise manual (and hence error prone) analysis of security issues relative to the business and
- 20 architectural context are enforced with a precise methodology which greatly reduces errors and omissions during the assessment process.

20

It should be appreciated that because the invention operates passively, the customer network can be monitored while in normal operation or production.

25

Operational Description

An example of implementing the assessment tool is described in the following discussion. A consultant arrives at a customer office with one or more workstations with the monitoring invention discussed herein loaded. The workstation, or station for short, may be a laptop computer, or other suitably portable platform. The monitoring station is attached to the customer network at a critical network bottleneck, *e.g.* just inside an Internet firewall, and monitors all traffic at that point in the network. From a security point of view, the monitoring station is entirely passive and invisible to the network. The monitoring station only receives packets and does not respond to any protocol actions. Due to the monitoring station's passive nature, no operational impact is imposed on the subject network. Hence, assessments may be performed during peak production times, as well as when a network is in a quiescent state.

In this example, the monitoring station is left attached to the network for a long period of time, depending on conditions, such as, for example, the practical demands of the visit, storage space on the station, and the amount of traffic on the customer's network. If appropriate, the station can be left at the customer site to gather data over a short-term period, such as, for example, days and weeks.

In this example of an assessment situation, the policy specification is used to remove from consideration as much mundane network traffic as possible, allowing the analyst to concentrate on more interesting traffic. Due to the opinion of the analyst being part of the assessment process, there is no fixed goal for the level of detail needed in the policy specification. In the simplest

case, the analyst generates no policy at all, and examines the network events one by one (perhaps using the query tool to filter them). In practice, it can be suggested that the analyst undergoes a short policy development phase, as the short policy development phase can serve the analyst well to reduce
5 thousands of network events into a page or two, which may then be examined by inspection.

The invention allows data to be stored in full packet form for most detailed analysis, or in compressed form storing only security-sensitive events. The
10 latter form also removes customer-confidential information, such as, for example, embedded passwords, so that it is more appropriate for removal from the customer site. A typical usage scenario is capturing full-packet data in a short burst, such as, for example, five minutes. After a brief analysis, a longer data collection is run using the compressed form.

15 The preferred embodiment of the invention provides the following algorithm for an operator, such as an analyst, to perform the data analysis on a data packet or on a compressed file of data. The algorithm is described referring to Fig. 14, as follows:

- 20
- 1) Create a null policy, which denies all actions, for a customer site (copying a file). Set null policy to the current policy (4002);
 - 2) Run the policy engine discussed herein over the input data and using
25 current policy (4002), and store the resulting data in a local database (4003);

- 3) Using the query tool discussed herein, examine the network traffic that is declared in violation by the current policy (4004);
- 5 4) Categorize the most frequent traffic based on customer input:
 - a) If the traffic matches known customer-supplied input patterns, add this traffic to the policy with an OK disposition (4005);
 - 10 b) If the traffic does not match customer-supplied input patterns, but has high volume, add this traffic to the policy with an OK,monitor disposition (4006) .
- 5) Repeat from step 2 (4009) until only a small, manageable number of
15 events remains (4007). Then end the algorithm (4008).

It should be appreciated that the same packet or compressed file is run by the policy engine multiple times.

- 20 It should be appreciated that in an assessment situation a policy can be edited by using the policy generator discussed herein. The invention provides for using the policy generator for rapid policy development based on transport-level parameters. Enhanced policy development, using more complex tools, typically is not necessary in an assessment situation.

It should also be appreciated implementing the algorithm discussed above does not take very long. Part or all of the process may take place at the customer site, in a hotel room, on an airplane, or back at the analyst's office, for example. When the process is completed, the analyst has a list of
5 monitored network events. This list is used as a basis for additional discussion with the customer to determine the meaning of such events. Experience has shown that such conversation is useful to the assessment interviewing process.

10 It should also be appreciated that the variations of the algorithm above can be implemented and are within the scope of the invention. Examples of variations follow.

Example Variation I

15 An equally preferred embodiment comprises the analysts first determining the customer requirements and the customer network credentials. Using this information, the analyst programs an initial policy. The analyst can derive and use additional information from the scanning process as described in the algorithm above.

20

Example Variation II

The customer or analysts designs an initial best policy as a set of credentials and rules, set all dispositions to DENY, and monitors the network to determine what the dispositions should be.

25

In the preferred embodiment of the invention, the policy language describes a policy decision involving two principals, an initiator and a target principal. These principals are identified by a set of one or more credentials. For each policy decision the policy engine ascertains which credential in the policy best describes the information about the principals involved in an interaction. Similarly, the policy language herein describes conditions that in turn describe tests performed on the state of an associated protocol event.

The preferred embodiment of the invention provides a credential / condition assertion verification optimization algorithm to ensure that the choice of credentials and conditions are made as efficiently as possible.

To accomplish credential / condition assertion verification optimization, the policy engine:

- During the initialization process dynamically creates comparing functions for principals with credentials, and comparing functions for state of protocol events with particular conditions in a high level language such as C++;
- Dynamically creates and loads a module containing the comparing functions;
- During runtime ensures that installed policy file matches module containing comparing functions, otherwise generates new module containing comparing functions that correspond to installed policy file; and

- Calls comparing functions as appropriate.

The preferred embodiment provides a more rigorous algorithm, an example of
5 which is described in Table M below.

Table M

During the initialization process of the policy engine:

- 10 ◦ the policy engine requests that the parser module load a policy file, comprising
 credentials and conditions into an in-memory representation;
- the policy engine requests that the parser module load an assertion verification
 dynamically loadable library (DLL);
- 15 ◦ if this DLL exists then
 - it is loaded into memory; and
 - 20 ◦ a predetermined function, for example named dllValidateFunc(), contained in
 the loaded DLL is called. If the return value of the function call is the same
 as a MD5 hash of the previously loaded policy file, then loading is complete.
 Otherwise execution initialization continues below;
- 25 ◦ because the DLL does not exist or because the MD5 hash does not match, a
 code-generation function of the parser module is invoked, which:
 - adds header information to a C++ assertion code file;
 - 30 ◦ adds a function that returns the MD5 hash of the policy file that was used to
 generate this C++ file;

- 5
 - iterates through credentials contained in the in-memory representation, generating C++ function prototype and function declarations for code that can compare a principal description with the definition of a credential into the assertion code file, wherein such comparison is performed by:
 - calling other credential comparison methods for any credentials used in the definition of the credential under test;
 - 10
 - making calls to the policy engine module to perform comparison operations based on allowable operations for the built-in types of the policy language; and
 - combining the results of the above tests with logical operators AND, OR and NOT;
- 15
 - iterates through the conditions contained in the in-memory representation, generating C++ function prototype and function declarations for code that can compare a protocol state description with the definition of a condition into the assertion code file, wherein such comparison is performed by:
 - calling other condition comparison methods for any conditions used in the definition of the condition under test;
 - 20
 - making calls to the policy engine module to perform comparison operations based on the allowable operations for the built-in types of the policy language; and
 - combining the results of the above tests with logical operators AND, OR and NOT;
- 25
 - making calls to the policy engine module to perform comparison operations based on the allowable operations for the built-in types of the policy language; and
 - combining the results of the above tests with logical operators AND, OR and NOT;
- 30
 - combining the results of the above tests with logical operators AND, OR and NOT;

- compiles and links this generated C++ file to create a dynamically loadable module containing a compiled version of the principal/credential and protocol/condition comparison functions; and

- 5
- loads this newly created module.

During the runtime of the policy engine:

- 10
- each time that it needs to decide whether a principal is described by a particular credential it computes the name of the comparison function based on the name of the credential to be tested;
 - calls the comparison function which returns a Boolean value that represents whether the credential under test matches the principal under test;

15

 - each time that it needs to decide whether a protocol state satisfies a particular condition it computes the name of the comparison function based on the name of the condition to be tested; and

20

 - calls the comparison function which returns a Boolean value that represents whether the condition under test satisfies the protocol state under test.

Network Monitor Internals Descriptions

The preferred embodiment of the invention provides a network monitor
internals mechanism discussed below that serves to translate packet data into
25 multiple concurrent streams of network event data. It accomplishes this by
interpreting both sides of each protocol transaction.

Fig. 15 shows a high level schematic diagram of the network monitor 127
accepting packet data from either a live network interface 125 or a file

containing packet data 126. The network monitor extracts security-sensitive details from the input packet stream 125, 126, and generates output in a serialized stream of encoded network event information 115. The preferred encoded format is DME encoded format, discussed below in section, Network

5 Event Encoding Format. The output network event information can be stored for logging or debugging purposes, or can be passed directly to the policy engine. Thus, the discussed network monitor provides an efficient process of exporting data from a customer's site, such process comprising extracting security-sensitive information.

10

Fig. 16 shows a schematic diagram of process flow according to the invention. The network monitor 127 is a single-threaded program that processes packets (125 or 126) as they are read. Each packet is passed to a monitor protocol engine 6100 for processing. When security-sensitive protocol events are

15 encountered in the packet data, the monitor calls into its output section 6200 to transmit network or protocol events to the rest of the policy monitoring system 100 via a network pipe, direct procedure call. Output section 6200 can also store protocol events in a file for later processing.

20 Protocol Engine

The preferred embodiment of the invention provides a protocol engine in the network monitor that can be described with reference to Figure 17, which is a block schematic diagram of features of the protocol engine according to the invention. Input packet data 115 is read into a known object-oriented

25 structure type 6101, such as, for example, a C structure here named pkt_t structure. The pkt_t structure 6101 represents a packet on the network. It

provides a stack-based structuring mechanism 6102 that allows protocol headers and trailers 6103 to be marked in the packet so that software may focus easily on the correct protocol layer. The `pkt_t` structure 6101 also includes generic `src` 6104 and `dst` 6105 address locations, and flags 6106 to

5 pass useful information up and down a connection stack, for example, if such packet is transiting from server to client or vice versa.

The protocol engine 6100 provides one module 6107 for each protocol implemented 6108. The modules implement a generic series of operations, a

10 preferred example of such series is provided below in Table N. A common connection structure 6109 allows connection data to be arranged in a stack allocation for each access across layer boundaries. In Java or C++ terminology, for example, each protocol is a superclass of connection. The layering permits protocols to assume one or more roles as the layer

15 responsible for each corresponding boundary, such as, for example: Network, Transport, Session, Application, or Transactions.

Table N

Example of generic operations for each protocol implementation:

- 20
1. Init: Call-once initialization
 2. Bind(packet, connection): given the first packet of a connection, attempt to bind this packet into a new instance of this protocol within *connection*. Establish the instance in its proper role(s) within the connection.
- 25

3. Input(packet, connection): given a packet, which has been associated with a connection (in some cases, connection is NULL, indicating that no such relationship exists, or exists yet), process the packet as input to the connection.
- 5 4. GiveBack(packet, connection): given a packet, which has been associated with a connection at a higher level of protocol, give back the packet to this layer, so that the data will be received later, as if it was retransmitted. Typically, *packet* has been modified to contain only part of the input data.
- 10 5. GetMore(connection, amountNeeded, fromClientOrServer) returns(packet): given a connection, attempt to return a packet containing more data on the connection, if such is available. This call is used from a higher layer of protocol calling down to a lower layer of protocol. The *fromClientOrServer* argument is used to determine if the data is being requested that was received by the server side or
- 15 the client side of the connection.
6. StopCollecting(connection): given a connection, adjust the protocol stack so that no further data will be processed on this connection. Depending on the protocol in question, this may involve discarding data or adjusting filters. A connection
- 20 which is not "collecting" attempts to process packets in the most efficient manner.
7. Shutdown(connection, fromOrg, fromDst): given a connection, modify the connection state to indicate that the client, server, or both have acted to take down the connection. The full generality of the call is needed only for a transport
- 25 connection like TCP.
8. Del(connection): given a connection, arbitrarily delete the instance of this protocol from the connection object. This call is intended to clean up the resources used by the connection; Shutdown is used to indicate protocol agreement that the
- 30 connection is coming to an end.

9. Alarm(connection, time): given a connection and the current time, this call is used to signal an alarm has expired on this connection. The time argument is the official time of the alarm, which may not even be related to the current time.

- 5 10. SwitchSrcDst(connection): this call indicates that a higher layer of software (perhaps a higher level protocol) has determined that the choice of client and server in this protocol instance are wrong, and should be reversed. This may happen when initial connection negotiation packets are not seen by the monitor, but later information makes the client and server clear.

10

It should be appreciated that in the stopCollecting generic operation, and in a transport protocol, header information in packets may need to be examined to determine connection state, allowing freeing of resources when the connection terminates. Transport protocols discard all subsequent data from the connection, and do not forward packets on to higher level protocols. Such mechanism allows the monitor to efficiently process bulk transfers, encrypted connections, or connections that are no longer of interest to the policy engine.

15

It should be appreciated that the process discussed above for the stopCollecting generic operation can be appropriate for a hardware filter to stop packets from arriving.

20

The concept of the current time in the monitor flows from the packet level upwards. That is, time is associated with the packet and is maintained throughout the packet. When the network monitor is running in real time off live packet data, current time reduces to the time a packet was received, which may be earlier than the time when the packet is processed. When the

25

network monitor is running off stored packet data, current time in the monitor has no relation to actual current time. The packet is processed relative to the time it was received and whereby time intervals remain the same. Also, results can be lined up in the database reflecting the point of reference of the
5 time the packet was received.

The network monitor provides support for setting alarms on connections. An alarm is set by registering a connection to receive a signal when the network monitor transitions to a predetermined value of current time. The signal
10 consists of a call to a generic alarm operation in every protocol layer registered with such connection. Alarm handlers are called in order from lowest protocol layer to highest protocol layer.

Because network monitor functionality is based on network events that can
15 map to network connections, the network monitor provides a connectionless association feature. By using the feature, the network monitor registers the fact that it noticed two IP hosts communicating. Typically, an association is long lived, whether or not the network monitor knows its intention. Examples of associations are a series of ICMP PING / PING REPLY packets and a
20 stream of IPSEC packets. The network monitor treats associations as connections. Indeed, often associations are connections at a higher level of protocol.

Output Section

25 The preferred embodiment of the invention provides an output section in the protocol engine. Fig. 18 is a high level flow diagram of the preferred output

section according to the invention. The output section 6200 of the network monitor receives network event data from the protocol engine and generates outbound calls 6203 to transmit such data to the policy engine or to a file.

- 5 The output section 6200 works by allowing the network monitor to establish a transaction which forms an association between a monitor connection and a network event in the policy engine. Fig. 19 shows a schematic diagram of a transaction 6204, comprising an association 6205 between a subject monitor connection 6206 and a network event 6207. Typically, the lifetime of the
- 10 connection 6206, the transaction 6204, and the network event 6207 is similar.

The output section's interface comprises a set of calls to establish communication with the policy engine, and to start and finish transactions, and a set of protocol-specific calls. The calls progress as follows:

15

- Connect
 - BeginTransaction
 - ProtocolEvent1
 - ProtocolEvent2
 - ...
 - EndTransaction
- Disconnect

20

- It should be appreciated that in addition to the calls above, multiple
- 25 transactions can be active at a time, as long as each transaction follows the ordering described above.

The output section internally translates such calls into a generic set of calls, an example of which is listed below. At initialization of the network monitor, the output section is configured with a chain of output generic modules, each of which is used as filter on the output data. An example of the implemented
5 modules follows:

- NULL: acts as an endpoint, but discards input data without doing anything;
- SM: connects by procedure call directly to policy processing;
- 10 ◦ ENC: generate encoded form of output; and
- LOG: generate textual form of output.

15 In an equally preferred embodiment of the invention, the network monitor also includes an input section that decodes an encoded version of events. For an example application, in a real-time monitoring system embodiment the monitor 127 processes network traffic 125 in real time and uses ENC to generate encoded output. The encoded output is transmitted in real-time over
20 a TCP connection where it is decoded and connected using SM to the policy engine 102.

In another embodiment of the invention, the output section is used for testing purposes. The output section is configured using command line arguments.

25 An example of an algorithm for such testing follows:

1. Capture packet data into a file;
2. Run the network monitor on the packet data, using LOG→ENC. Store the logged textual data and the encoded form into separate files; and

5

3. Run the network monitor on the encoded data, using LOG→NULL. Store the logged textual data in a file.

4. Compare the two textual files to make sure that the decoded version matches the logged textual file.

10

Network Event Encoding Format

The preferred embodiment of the invention provides a technique for network event encoding to be used by the network monitor. The encoding technique is designed for both archival and transmission purposes. The basic format of the encoding is:

15

- Header
- Embedded agent descriptors
- Type map
- Encoded transactions

20

An example of the preferred form of the header follows:

- 4 byte magic number: "SMKo"
- 1 byte major version = 2
- 1 byte minor version = 1
- 4 bytes containing the size of this header

25

- 8 bytes (struct timeval) begin time, which is a time which is less than or equal to every timestamp in this encoded record
- 4 bytes offset of agent descriptor section
- 4 bytes indicating number of agent descriptors
- 5 ◦ 4 bytes offset of type map section
- 4 bytes indicating number of type map entries
- 4 bytes offset to first transaction record
- 4 bytes size of this file, or 0xFFFFFFFF if unknown.
- 4 bytes 1's complement checksum of this file or 0xFFFFFFFF if unknown

10

The agent descriptor section is used to store a possibly null list of agent descriptors that are configured into the network monitor at encoding time.

The agent descriptors are strings that plug into a particular policy language policy. They indicate the location of the subject monitor in the subject network

15 wiring structure, enabling rules that apply to such location in the network and disable rules that do not apply.

A preferred agent descriptor section comprises an array, where each element of the array is an ASCII string, preceded by a single byte giving its length.

20 The size of the array is given in the header cited above.

The preferred type map section is used to improve maintainability of the full policy monitoring system. Provided by the type map section is a mapping between update types used in an encoded record and the update types' string

25 names. The decoding module uses this information to detect new update types that are not supported by mapping known updates to the correct values.

That is, because new update types typically are not interpretable by old software, they are therefore successfully skipped.

5 A preferred type map section comprises an array, where each element of the array contains a four-byte type value, a single byte of string length, and the ASCII name of the type. The size of the array is given in the header cited above.

10 The preferred encoded transactions comprise an array of individual update encodings. The size of the array is either derivable from the header file size information, or is unbounded, such as, for real-time monitoring.

A preferred header for an individual update has the following format:

- 15
- 1 byte, giving the update type
 - 4 bytes, giving the size of this header in bytes, not including the length of the header
 - 8 bytes (struct timeval) giving the absolute time when this update occurred
 - 4 bytes, giving the packet number of this update since the monitor started (first packet = packet #0)
- 20
- 4 bytes, giving the eventID of this update, which is the number of BEGIN_TRANS updates that occurred before this one, since the monitor started

Following the header a body contains additional update-type-specific data, or possibly none.

25

To understand all events that transpire on a connection, it is necessary to combine events of different protocol layers. For example, an update, named

SM_IP_ASSOCIATION, provides IP src and dst addresses and establishes a peer relationship. Subsequent events assume that this information is known and builds on it. For example, an update named ICMP_ECHO has no body at all.

5

An example of a set of update types and corresponding encoding body for each update, according to the invention is given below in Table O. The meaning of the term "string" is: *if length(string) is < 255, then byte[length], byte[string][length], else byte[0xff], byte[a], byte[b], byte[c], byte[d],*

10 *byte[string][length]* where a,b,c,d are the four (big-endian) bytes of length.

Table O

SM_BEGIN_TRANS

Body: none

15

Meaning: begin new transaction (network event)

SM_END_TRANS

Body: none

Meaning: end previously "begin" transaction (network event)

20

SM_PUOSU

Body: none

Meaning: the monitor can glean no more useful information about this network event. The policy engine should process policy and give additional input to the monitor.

25

SM_DEBUG_MSG

Body: string

Meaning: debug message, to be inserted into SPM debugging log.

SM_PROTOCOL_UNKNOWN

Body: none

Meaning: the monitor is unable to determine the higher level protocol

5

SM_FTP_DATAOPEN

Body: none

Meaning: This (new) connection is an FTP data connection

SM_FTP_DATACLOSE

10

Body: none

Meaning: This FTP data connection has closed normally.

SM_FTP_DATAABORT

Body: none

15

Meaning: This FTP data connection has close abnormally.

SM_FTP_OPEN

Body: none

Meaning: This (new) connection is an FTP control connection

20

SM_FTP_CLOSE

Body: none

Meaning: This FTP control connection has closed normally.

SM_FTP_ABORT

25

Body: none

Meaning: This FTP control connection has closed abnormally

SM_FTP_NOAUTH

Body: 4-byte, number of authentication failures

30

Meaning: This FTP control connection has failed to authenticate

SM_FTP_AUTH

Body: String, user name

String, password, if user was *anonymous*

4-byte, password length

5 1-byte, nonzero if password contains alphabetic

1-byte, nonzero if password contains numeric characters

1-byte, nonzero if password contains characters which are non-alphanumeric

4-byte, number of authentication failures

10 Meaning: This FTP control connection has successfully authenticated

SM_FTP_FILEGET

SM_FTP_FILEPUT

SM_FTP_DEL

15 SM_FTP_MKDIR

SM_FTP_RMDIR

Body: String, file name

1-byte, FTP error code

String, FTP error message

20 Meaning: attempt to perform FTP RETR, STORE, DEL, MKD, RMD command. If immediate failure, the error is given in the message. For GET/PUT, if transfer is proceeding, error status comes in the XFERDONE message.

SM_FTP_XFERDONE

25 Body: String, unused

1-byte, FTP error code

String, FTP error message

Meaning: status from continuing FILEPUT or FILEGET command

30 SM_FTP_RENAME

Body: String, *from* file name

String, *from* file name

1-byte, FTP error code

String, FTP error message

Meaning: attempt to perform FTP file rename command. If failure, the error is given in the message.

5

SM_HTTP_CLOSE

Body: none

Meaning: This HTTP connection has closed normally.

10

SM_HTTP_METHOD

Body: 1-byte, method code (one value for each HTTP method)

1-byte, HTTP version (major)

1-byte, HTTP version (minor)

15

String, URL

Meaning: Describes HTTP method line

SM_HTTP_POSTDATA

Body: 1-byte, always true.

20

1-byte, nonzero if this is the last POSTDATA call to complete all the post data.

String, post data

Meaning: contains some or all of the post data for an HTTP POST method.

25

SM_HTTP_REQCTYPE

SM_HTTP_RESPCTYPE

Body: String, content type

Meaning: HTTP content type from request or response header.

30

SM_HTTP_REQCOOKIE

SM_HTTP_RESPSETCOOKIE

Body: String

Meaning: HTTP cooking / set-cookie headers

SM_HTTP_REQHEADER

5

SM_HTTP_RESPHEADER

Body: 1-byte, nonzero if this is the last group of header info

4-byte, number of header lines

String[number of header lines]

Meaning: contains HTTP header information from request or response header.

10

SM_HTTP_REQHEADEREND

SM_HTTP_RESPHEADEREND

Body: none

Meaning: End of request or response header has been reached.

15

SM_HTTP_RESPONSE

Body: 4-byte, response code

1-byte, HTTP version (major)

1-byte, HTTP version (minor)

20

String, response message

Meaning: encoding of the HTTP response header line

SM_HTTP_MISS

Body: none

25

Meaning: Monitor was unable to parse the HTTP transaction (perhaps because of missed packets)

SM_ICMP_BADCODE

Body: none

30

Meaning: ICMP packet received of unknown type

- SM_ICMP_DU_FRAG (destination unreachable: fragmentation needed and DF set)
- SM_ICMP_DU_HOST (destination unreachable: host unreachable)
- SM_ICMP_DU_NET (destination unreachable: net unreachable)
- SM_ICMP_DU_PORT (destination unreachable: port unreachable)
- 5 SM_ICMP_DU_PROT (destination unreachable: protocol unreachable)
- SM_ICMP_DU_SRCRT (destination unreachable: source route failed)
- SM_ICMP_DU_FILTER (destination unreachable: packet filtered)
- SM_ICMP_PARAM (parameter problem)
- SM_ICMP_SRCQ (source quench)
- 10 SM_ICMP_TE_EXCD (time to live exceeded in transit)
- SM_ICMP_TE_FRAG (fragment reassembly time exceeded)
- Body: 4-byte, IP src address
- 2-byte, UDP/TCP src port
- 4-byte, IP dst address
- 15 2-byte, UDP/TCP src port
- 4-byte, IP protocol
- Meaning: This connection contains a particular ICMP error. The body gives information from the nested packet within the ICMP packet.
- 20 SM_ICMP_ECHO
- SM_ICMP_ECHOR
- Body: none
- Meaning: ICMP echo / echo reply seen (echo is commonly called "ping").
- 25 SM_ICMP_IREQ
- SM_ICMP_IREQR
- Body: none
- Meaning: ICMP information request/reply seen
- 30 SM_ICMP_RD_HOST (Redirect datagrams for the Host)
- SM_ICMP_RD_HOSTTOS (Redirect datagrams for the Type of Service and Host)

SM_ICMP_RD_NET (Redirect datagrams for the Network)

SM_ICMP_RD_NETTOS (Redirect datagrams for the Type of Service and Network)

Body: 4-byte, gateway address

4-byte, IP src address

5 2-byte, UDP/TCP src port

4-byte, IP dst address

2-byte, UDP/TCP src port

4-byte, IP protocol

10 Meaning: For the given ICMP redirect, the body gives gateway information and information from the nested packet within the ICMP packet.

SM_ICMP_TSTMP

SM_ICMP_TSTMPR

Body: none

15 Meaning: ICMP Timestamp / Timestamp reply seen

SM_ICMP_ASSOCIATION

Body: none

Meaning: This connection contains an ICMP-level association.

20

SM_IPINFO_IP_ASSOCIATION

Body: 6-byte, src MAC address

6-byte, dst MAC address

4-byte, IP src address

25 2-byte, UDP/TCP src port

4-byte, IP dst address

2-byte, UDP/TCP src port

1-byte, IP protocol

1-byte, IP version

30 Meaning: an IP protocol association exists on this connection.

SM_TCP_CONNECT

SM_TCP_MISSED_CONNECT

Body: none

Meaning: a (new) TCP connection exists on this connection. In the case of a "missed" connect, the first packets from the connection were not seen, so the monitor is unable to properly classify the connection.

5

SM_TCP_DATA

Body: none

Meaning: data has transited this connection

10

SM_UDP_ASSOCIATION

Body: none

Meaning: This connection contains a (new) UDP association

15

SM_SSH_AUTH

Body: 4-byte, client version (major)

4-byte, client version (minor)

4-byte, server version (major)

4-byte, server version (minor)

4-byte, authmask, gives which cipher suites are supported (see SSH specification)

4-byte, cipher suite selected

Meaning: a successful SSH authentication has occurred.

25

SM_SSH_ABORT

SM_SSH_CLOSE

Body: none

Meaning: the SSH connection has terminated. An ABORT means that the transport layer aborted.

30

SM_SSH_HANDSHAKE_FAILURE

Body: none

Meaning: the monitor was able to determine that the SSH handshake failed.

5 SM_SSH_HANDSHAKE_MISS, // We cannot interpret the handshake.

Body: none

Meaning: the monitor was unable to determine whether the SSH handshake failed or succeeded.

10 SM_SSL_ABORT (fatal alert)

SM_SSL_WARNING (non-fatal alert)

SM_SSL_HANDSHAKE_FAILURE (alert seen, indicates handshake failure)

Body: 1-byte, alert level (see SSL3 specification)

1-byte, alert description

15 Meaning: The SSL connection has signaled an ALERT.

SM_SSL_HANDSHAKE_SUCCEED

Body: none

Meaning: the SSL connection has completed its handshake

20

SM_SSL_HANDSHAKE_ABORT

Body: none

Meaning: the SSL connection was aborted by transport level without handshake completion

25

SM_SSL_HANDSHAKE_MISS

Body: none

Meaning: The monitor was unable to determine the SSL session credentials. Because of resumed sessions, this may mean that the session was completely successful.

30

SM_SSL_SERVER_HELLO

Body: 1-byte, version (major)

1-byte, version (minor)

4-byte, ciphersuite (enum)

5 1-byte, non-zero if a resumed session

String, sessionid

Meaning: SSL (client+)server hello information

SM_SSL_CLIENT_CERT

10 SM_SSL_SERVER_CERT

Body: String, client or server certificate chain

Meaning: client or server certificate

SM_TCP_ABORT

15 Body: none

Meaning: TCP RST packet received, killed connection

SM_TCP_CLOSE

Body: none

20 Meaning: TCP normal close (both sides)

SM_TCP_TIMEOUT

Body: none

Meaning: TCP death timer expires, killing connection.

25

Table R

(policy PolicyGen "0.9"

30

(group PolicyGen_Monitors agent_attr_t

(union

X_MONITOR

)

)

5

(credential Home_Machine

(assertion

(eq ip-address 10.0.0.176)) // assertion

)

10

(credential Cgi

(assertion

(eq ip-address 10.0.0.119)) // assertion

)

15

(credential Clouds

(assertion

(eq ip-address 10.0.0.118)) // assertion

20

)

(credential Fluffy

(assertion

25

(eq ip-address 10.0.0.125)) // assertion

)

(credential Monkey

30

(assertion

(or

```
( eq ip-address 10.0.0.114 )
( eq ip-address 10.0.0.115 )
( eq ip-address 10.0.0.121 )
    ) // or
5      ) // assertion
      )

( credential X_Web_Servers
10      ( assertion
        ( or
          Cgi
          Clouds
          Fluffy
15      Monkey
        ) // or
      ) // assertion
    )

20
( credential Security_Web_Server
  ( assertion
    ( eq ip-address 10.0.0.120 ) ) // assertion
  )
25

( credential All_Web_Servers
  ( assertion
    ( or
30      X_Web_Servers
      Security_Web_Server
```

```
        ) // or
    ) // assertion
)
```

5

```
( credential Anon_User
  ( assertion
    ( or
      ( eq login-name "anonymous" )
    ) // or
  ) // assertion
)
```

10

15

```
( credential Dns_Server
  ( assertion
    ( eq ip-address 10.0.0.21 ) ) // assertion
)
```

20

```
( credential Ip_Directed_Broadcasts_Within_X_Target
  ( assertion
    ( or
      ( eq ip-address 10.0.0.119 )
    ) // or
  ) // assertion
)
```

25

30

```
( credential X_Coloc_Subnet
  ( assertion
```

```
( ip-mask ip-address 10.0.0.112/29 ) ) // assertion
```

```
)
```

```
5      ( credential __Zero_Ip_Address
      ( assertion
        ( eq ip-address 10.0.0.0 ) ) // assertion
      )
```

```
10      ( credential Ip_Within_X_Initiator
      ( assertion
        ( or
          X_Coloc_Subnet
15      __Zero_Ip_Address
        ) // or
      ) // assertion
    )
```

```
20      ( credential Loghost
      ( assertion
        ( eq ip-address 10.0.0.190 ) ) // assertion
      )
```

```
25      ( credential Modin
      ( assertion
        ( eq ip-address 10.0.0.117 ) ) // assertion
30      )
```

```
( credential Mother
  ( assertion
    ( eq ip-address 10.0.0.124 ) ) // assertion
5  )

( credential X_Netops
  ( assertion
10    ( ip-range ip-address 10.0.0.187 10.0.0.190 ) ) // assertion
  )

( credential Security
15  ( assertion
    ( eq ip-address 10.0.0.61 ) ) // assertion
  )

20  ( credential X_External_Communities
    ( assertion
      ( or
        Home_Machine
        Dns_Server
25        Loghost
        X_Netops
        Security
      ) // or
    ) // assertion
30  )
```

```
( credential X_Monitors
```

```
  ( assertion
```

```
    ( member X_MONITOR agent-attribute )    ) // assertion
```

```
5      )
```

```
( credential X_Ssh_From_X_To_X_Web_Servers_Provisional_Initiator
```

```
  ( assertion
```

```
10    ( or
```

```
      Home_Machine
```

```
      X_Netops
```

```
    ) // or
```

```
  ) // assertion
```

```
15    )
```

```
( credential X_Ssh_From_X_To_X_Web_Servers_Provisional_Target
```

```
  ( assertion
```

```
20    ( or
```

```
      Mother
```

```
      X_Web_Servers
```

```
    ) // or
```

```
  ) // assertion
```

```
25    )
```

```
( credential X_Ssh_To_Security_Web_Server_Initiator
```

```
  ( assertion
```

```
30    ( or
```

```
      X_Netops
```

Security

) // or

) // assertion

)

5

(credential X_Stmp_From_All_To_X_Initiator

(assertion

(or

10

Cgi

Clouds

) // or

) // assertion

)

15

(credential _Dns

(assertion

(eq ip-port 53)) // assertion

20

)

(credential Tcp_X_Dns_From_Colloc_To_Dns_Server_Target

(assertion

25

(and

Dns_Server

_Dns

) // and

) // assertion

30

)

```
( credential _Http
  ( assertion
    ( eq ip-port 80 ) ) // assertion
```

```
5      )
```

```
( credential Tcp_X_Http_From_Any_To_All_Web_Servers_Provisional_Target
```

```
  ( assertion
```

```
10      ( and
```

```
        All_Web_Servers
```

```
        _Http
```

```
      ) // and
```

```
    ) // assertion
```

```
15      )
```

```
( credential _Bigbrother
```

```
  ( assertion
```

```
20      ( eq ip-port 1984 ) ) // assertion
```

```
  )
```

```
( credential Tcp_X_Port_1984_Traffic_Target
```

```
25      ( assertion
```

```
        ( and
```

```
          Loghost
```

```
          _Bigbrother
```

```
        ) // and
```

```
30      ) // assertion
```

```
    )
```

```
( credential _Ssh26
  ( assertion
5      ( eq ip-port 26 ) ) // assertion
  )

( credential Tcp_X_X_Loghost_Traffic_Target
10  ( assertion
    ( and
      Loghost
      _Ssh26
    ) // and
15  ) // assertion
  )

( credential _Ssh
20  ( assertion
    ( eq ip-port 22 ) ) // assertion
  )

25  ( credential Tcp_X_Shsh_From_Clouds_To_Cgi_Provisional_Target
    ( assertion
      ( and
        Cgi
        _Ssh
30      ) // and
    ) // assertion
```

)

(credential Tcp_X_Spm_Colloc_Traffic_Provisional_Target

5

(assertion

(and

Security

_Ssh

) // and

10

) // assertion

)

(credential _Sntp

15

(assertion

(eq ip-port 25)) // assertion

)

20

(credential Tcp_X_Spm_Colloc_Traffic_Target

(assertion

(and

Security

_Sntp

25

) // and

) // assertion

)

30

(credential Tcp_X_Ssh_From_Fluffy_To_Monkey_Provisional_Target

(assertion

```
( and
    Monkey
    _Ssh
) // and
5    ) // assertion
)

( credential Tcp_X_Ssh_From_Monkey_To_Fluffy_Provisional_Target
10    ( assertion
        ( and
            Fluffy
            _Ssh
        ) // and
15    ) // assertion
    )

( credential Tcp_X_Ssh_From_X_To_X_Web_Servers_Provisional_Target
20    ( assertion
        ( and
            X_Ssh_From_X_To_X_Web_Servers_Provisional_Target
            _Ssh
        ) // and
25    ) // assertion
    )

( credential _Ssh20
30    ( assertion
        ( eq ip-port 20 ) ) // assertion
```

)

(credential Tcp_X_Ssh_To_Security_Web_Server_Target

5

(assertion

(and

Security_Web_Server

_Ssh20

) // and

10

) // assertion

)

(credential Udp_X_Dns_From_Colloc_To_Dns_Server_Target

15

(assertion

(and

Dns_Server

_Dns

) // and

20

) // assertion

)

(credential _Auth

25

(assertion

(eq ip-port 113)) // assertion

)

30

(credential _Bootp_Client

(assertion

```
( eq ip-port 68 )) // assertion
```

```
)
```

```
5      ( credential _Bootp_Server
```

```
      ( assertion
```

```
        ( eq ip-port 67 )) // assertion
```

```
)
```

```
10
```

```
      ( credential _Finger
```

```
      ( assertion
```

```
        ( eq ip-port 79 )) // assertion
```

```
)
```

```
15
```

```
      ( credential _Ftp
```

```
      ( assertion
```

```
        ( eq ip-port 21 )) // assertion
```

```
20
```

```
)
```

```
      ( credential _Gopher
```

```
      ( assertion
```

```
25
```

```
        ( eq ip-port 70 )) // assertion
```

```
)
```

```
      ( credential _High_Ports
```

```
30
```

```
      ( assertion
```

```
        ( range ip-port 1025 65535 ) ) // assertion
```

)

```
5      ( credential _Https
        ( assertion
          ( eq ip-port 443 ) ) // assertion
        )
```

```
10     ( credential _Ident
        ( assertion
          ( eq ip-port 113 ) ) // assertion
        )
```

```
15     ( credential _Imap4
        ( assertion
          ( eq ip-port 143 ) ) // assertion
        )
```

```
20     ( credential _Imap4s
        ( assertion
          ( eq ip-port 993 ) ) // assertion
25     )
```

```
      ( credential _Netbios_Rpc
        ( assertion
30      ( eq ip-port 135 ) ) // assertion
        )
```

```
( credential _Nntp
  ( assertion
5      ( eq ip-port 119 ) ) // assertion
  )

( credential _Pop3
10   ( assertion
      ( eq ip-port 110 ) ) // assertion
  )

15   ( credential _Pop3s
      ( assertion
          ( eq ip-port 995 ) ) // assertion
      )

20   ( credential _Printer
      ( assertion
          ( eq ip-port 515 ) ) // assertion
      )

25   ( credential _Rlogin
      ( assertion
          ( eq ip-port 513 ) ) // assertion
30   )
```

```
( credential _Rshell
  ( assertion
    ( eq ip-port 514 ) ) // assertion
5  )

( credential _Smb
  ( assertion
10  ( range ip-port 137 139 ) ) // assertion
  )

( credential _Smtps
15  ( assertion
    ( eq ip-port 465 ) ) // assertion
  )

20  ( credential _Syslog
    ( assertion
      ( eq ip-port 514 ) ) // assertion
    )

25  ( credential _Telnet
    ( assertion
      ( eq ip-port 23 ) ) // assertion
    )
30
```

```
( credential _Whois
  ( assertion
    ( eq ip-port 43 ) ) // assertion
  )
```

5

```
( credential __Multicast_Addresses
  ( assertion
    ( ip-range ip-address 224.0.0.0 239.255.255.255 ) ) // assertion
  )
```

10

```
( credential __Non_Directed_Broadcast_Address
  ( assertion
    ( and
      ( eq ip-address 255.255.255.255 )
      ( eq mac-address FF-FF-FF-FF-FF-FF )
    ) // and
  ) // assertion
  )
```

15

20

```
( credential __Generic_Multicast_And_Broadcast_Addresses
  ( assertion
    ( or
      __Non_Directed_Broadcast_Address
      __Multicast_Addresses
    ) // or
  ) // assertion
  )
```

25

30

```
( condition Authentication_Rejected
  ( assertion
    ( eq auth-status REJECTED ) ) // assertion
5  )

( condition Ssh_Authentication_Aborted
  ( assertion
10  ( eq ssh-handshake-status ABORTED ) ) // assertion
  )

( condition Ssh_Authentication_Failed
15  ( assertion
    ( eq ssh-handshake-status FAILED ) ) // assertion
  )

20  ( condition Ssh_Secure_Authentication_Modes
    ( assertion
      ( or      ( member
                  SSH_RSA ssh-supported-auth-modes )      ( member
                  SSH_RHOSTS_WITH_RSA ssh-supported-auth-modes )
25      ) // or
    ) // assertion
  )

30  ( condition Ssl_Session_Qos
    ( assertion
```

```
( and
  ( or
    ( absent initiator-auth-keysize )
    ( ge initiator-auth-keysize 1024 )
  ) // or

  ( ge target-auth-keysize 1024 )
  ( ge ke-keysize 768 )
  ( ge encipher-keysize 128 )
  ( ge protocol-version ( version "3.0" ) )
) // and
) // assertion
)

( condition Strong_Password
  ( assertion
    ( and
      ( ge password-length 8 )
      ( or
        ( eq password-has-alphabetic true )
        ( eq password-has-numeric true )
      ) // or
      ( eq password-has-special true )
    ) // and
  ) // assertion
)

( condition Tcp_Data_Xfer
```

```
( assertion
  ( eq tcp-data true ) ) // assertion
)
```

5

```
( disposition Authentication_Failed
  ( code AUTHENTICATION_VIOLATION )
  ( log-directive
    HIGH
    "Authentication handshake failed"
  )
)
```

10

15

```
( disposition Ftp_Access_Violation
  ( code ACCESS_DENIED )
  ( log-directive
    HIGH
    "Illegal traffic at FTP level"
  )
)
```

20

```
( disposition Handshake_Aborted
  ( code AUTHENTICATION_VIOLATION )
  ( log-directive
    INFORMATION
    "Authentication handshake aborted by either party"
  )
)
```

25

30

```
( disposition Http_Access_Violation
  ( code ACCESS_DENIED )
```

```
( log-directive
    HIGH
    "Illegal traffic at HTTP level"
)
5      )

( disposition Icmp_Access_Violation
    ( code ACCESS_DENIED )
    ( log-directive
10      HIGH
        "Illegal traffic at ICMP level"
    )
)

15      ( disposition Incorrect_Port_Usage
        ( code SECURITY_ATTACK )
        ( log-directive
            MEDIUM
            "A TCP/UDP service is being used by an unexpected/unknown protocol"
20      )
    )

( disposition Ip_Access_Violation
    ( code ACCESS_DENIED )
25      ( log-directive
        HIGH
        "Illegal traffic at IP level"
    )
)

30      ( disposition Monitor_Anonymous_Login
```

(code OK)

(log-directive

MONITOR

"Anonymous login is being used"

5

)

)

(disposition Monitor_Broadcasts

(code OK)

10

(log-directive

MONITOR

"Multicast or Broadcast traffic detected"

)

)

15

(disposition Monitor_Icmp

(code OK)

(log-directive

MONITOR

20

"ICMP traffic detected"

)

)

(disposition Probable_Scan

25

(code SECURITY_ATTACK)

(log-directive

WARNING

"A probable network scan of a blocked TCP service has been detected"

)

30

)

```
( disposition Protocol_Unknown
  ( code ACCESS_DENIED )
  ( log-directive
    HIGH
5    "A protocol not understood by the monitoring system has been detected"
  )
)

( disposition Ssh_Access_Violation
10  ( code ACCESS_DENIED )
    ( log-directive
      HIGH
      "Illegal traffic at SSH level"
    )
15  )

( disposition Ssl_Access_Violation
    ( code ACCESS_DENIED )
    ( log-directive
20      HIGH
      "Illegal traffic at SSL level"
    )
    )

25  ( disposition Tcp_Access_Violation
    ( code ACCESS_DENIED )
    ( log-directive
      HIGH
      "Illegal traffic at TCP level"
30  )
    )
)
```

```
( disposition Udp_Access_Violation
  ( code ACCESS_DENIED )
  ( log-directive
5      HIGH
      "Illegal traffic at UDP level"
  )
)

10 ( disposition Warn_Missed_Handshake
    ( code OK )
    ( log-directive
        WARNING
        "Missed the authentication handshake"
15    )
    )

    ( disposition Warn_Missed_Tcp_Connect
        ( code OK )
20        ( log-directive
            WARNING
            "Missed TCP connect"
        )
    )

25
    ( disposition Weak_Authentication
        ( code SECURITY_QOS )
        ( log-directive
            HIGH
30        "A weak authentication mode or mechanism is being allowed"
        )
    )
```

```
)

( disposition Weak_Password
  ( code SECURITY_QOS )
5    ( log-directive
      HIGH
      "A weak password is being used for authentication"
    )
  )
10 )

( rule Ftp_Anonymous_Authentication
  ( protocol FTP )
  ( action CONTROL_AUTHENTICATE )
  ( initiator Anon_User )
15  ( target ignore )
    ( outcome
      ( immediate
        ( if Authentication_Rejected Authentication_Failed )
        ( default Monitor_Anonymous_Login )
20      )
    )
  )
  )

( rule Tcp_Blocked_Services
25  ( protocol TCP )
    ( action CONNECT )
    ( initiator ignore )
    ( target ignore )
    ( outcome
30      ( final
        ( default Probable_Scan )
      )
    )
  )
```

)
)
)

5 (rule Ftp_Blocked_Service_Violation
 (protocol FTP)
 (action ignore)
 (prerequisite Tcp_Blocked_Services)
 (initiator ignore)
10 (target ignore)
 (outcome
 (immediate
 (default Ftp_Access_Violation)
)
)
15)
)
)
 (rule Ftp_Deny
 (protocol FTP)
20 (action ignore)
 (initiator ignore)
 (target ignore)
 (outcome
 (immediate
25 (default Ftp_Access_Violation)
)
)
)
)
)
)
30 (rule Ftp_Ignore_Data_Connections
 (protocol FTP)

```
( action DATA_OPEN )
( initiator ignore )
( target ignore )
( outcome
5      ( immediate
      ( default ok )
      )
    )
  )
)

10

( rule Ftp_Validate_Password
  ( protocol FTP )
  ( action CONTROL_AUTHENTICATE )
  ( initiator ignore )
15  ( target ignore )
  ( outcome
    ( immediate
      ( if Authentication_Rejected Authentication_Failed )
      ( ifnot Strong_Password Weak_Password )
20      ( default ok )
    )
  )
)

25

( rule Http_Blocked_Service_Violation
  ( protocol HTTP )
  ( action ignore )
  ( prerequisite Tcp_Blocked_Services )
  ( initiator ignore )
30  ( target ignore )
  ( outcome
```

```

        ( immediate
          ( default Http_Access_Violation )
        )
      )
5    )

    ( rule Http_Deny
      ( protocol HTTP )
      ( action ignore )
10   ( initiator ignore )
      ( target ignore )
      ( outcome
        ( immediate
          ( default Http_Access_Violation )
15   )
        )
      )

    ( rule Icmp_Deny
20   ( protocol ICMP )
      ( action ignore )
      ( initiator ignore )
      ( target ignore )
      ( outcome
25   ( immediate
        ( default Icmp_Access_Violation )
      )
    )
  )
30

  ( rule Ip_Within_X
```

```
( protocol IP )
( action ASSOCIATION )
( agent X_Monitors )
( initiator Ip_Within_X_Initiator )
5 ( target X_Coloc_Subnet )
( outcome
  ( final
    ( default Protocol_Unknown )
  )
10 )
)

( rule Icmp_Within_X
  ( protocol ICMP )
15 ( action ASSOCIATION )
  ( agent X_Monitors )
  ( prerequisite Ip_Within_X )
  ( initiator ignore )
  ( target ignore )
20 ( outcome
    ( immediate
      ( default Monitor_Icmp )
    )
  )
25 )

( rule Ip_Deny
  ( protocol IP )
  ( action ignore )
30 ( initiator ignore )
  ( target ignore )
```

```
( outcome
  ( immediate
    ( default Ip_Access_Violation )
  )
)
```

```
5      )
      )
```

```
( rule Ip_Directed_Broadcasts_Within_X
  ( protocol IP )
10  ( action ASSOCIATION )
  ( agent X_Monitors )
  ( initiator Ip_Within_X_Initiator )
  ( target Ip_Directed_Broadcasts_Within_X_Target )
  ( outcome
15    ( immediate
      ( default Monitor_Broadcasts )
    )
  )
20 )
```

```
( rule Ip_External_Communities_To_X
  ( protocol IP )
  ( action ASSOCIATION )
  ( agent X_Monitors )
25 ( initiator X_External_Communities )
  ( target X_Coloc_Subnet )
  ( outcome
    ( final
      ( default Protocol_Unknown )
30    )
  )
)
```

)

(rule Ip_Non_Directed_Broadcasts_Within_X

(protocol IP)

5 (action ASSOCIATION)

(agent X_Monitors)

(initiator Ip_Within_X_Initiator)

(target __Generic_Multicast_And_Broadcast_Addresses)

(outcome

10 (immediate

(default Monitor_Broadcasts)

)

)

)

15

(rule Ip_X_To_External_Communities

(protocol IP)

(action ASSOCIATION)

(agent X_Monitors)

20 (initiator X_Coloc_Subnet)

(target X_External_Communities)

(outcome

(final

(default Protocol_Unknown)

25)

)

)

(rule Ip_Unknown_Protocol

30 (protocol IP)

(action PROTOCOL_UNKNOWN)

```
( initiator ignore )
( target ignore )
( outcome
  ( immediate
    ( default Protocol_Unknown )
  )
)
)
)

10 ( rule Ssh_Blocked_Service_Violation
    ( protocol SSH )
    ( action ignore )
    ( prerequisite Tcp_Blocked_Services )
    ( initiator ignore )
15 ( target ignore )
    ( outcome
      ( immediate
        ( default Ssh_Access_Violation )
      )
20 )
)

( rule Ssh_Deny
    ( protocol SSH )
25 ( action ignore )
    ( initiator ignore )
    ( target ignore )
    ( outcome
      ( immediate
30 ( default Ssh_Access_Violation )
      )
    )
)
```

```
)  
)  
  
( rule Ssh_Validate_Handshake  
5      ( protocol SSH )  
      ( action ( union HANDSHAKE SESSION_ABORTED ) )  
      ( initiator ignore )  
      ( target ignore )  
      ( outcome  
10      ( immediate  
          ( if Ssh_Authentication_Failed Authentication_Failed )  
          ( if Ssh_Authentication_Aborted Handshake_Aborted )  
          ( ifnot Ssh_Secure_Authentication_Modes Weak_Authentication )  
          ( default ok )  
15      )  
      )  
    )  
  
( rule Ssl_Blocked_Service_Violation  
20      ( protocol SSL )  
      ( action ignore )  
      ( prerequisite Tcp_Blocked_Services )  
      ( initiator ignore )  
      ( target ignore )  
25      ( outcome  
          ( immediate  
              ( default Ssl_Access_Violation )  
          )  
      )  
30      )  
    )
```

```
( rule Ssl_Deny
  ( protocol SSL )
  ( action ignore )
  ( initiator ignore )
5   ( target ignore )
    ( outcome
      ( immediate
        ( default Ssl_Access_Violation )
      )
    )
10  )
   )

( rule Ssl_Missed_Handshakes
  ( protocol SSL )
15  ( action MISSED_HANDSHAKE )
    ( initiator ignore )
    ( target ignore )
    ( outcome
      ( immediate
20      ( default Warn_Missed_Handshake )
      )
    )
  )

25  ( rule Ssl_Validate_Handshake
    ( protocol SSL )
    ( action HANDSHAKE )
    ( initiator ignore )
    ( target ignore )
30  ( outcome
      ( immediate
```

```
( if Authentication_Rejected Authentication_Failed )  
( ifnot Ssl_Session_Qos Weak_Authentication )  
( default ok )
```

```
)
```

```
5      )  
      )
```

```
( rule Tcp_Blocked_Services_Response
```

```
( protocol TCP )
```

```
10    ( action ( union ABORT CLOSE TIMEOUT ) )
```

```
( prerequisite Tcp_Blocked_Services )
```

```
( initiator ignore )
```

```
( target ignore )
```

```
( outcome
```

```
15    ( immediate
```

```
( if Tcp_Data_Xfer Tcp_Access_Violation )
```

```
( default Probable_Scan )
```

```
)
```

```
)
```

```
20    )
```

```
( rule Tcp_Blocked_Services_Violation
```

```
( protocol TCP )
```

```
( action PROTOCOL_UNKNOWN )
```

```
25    ( prerequisite Tcp_Blocked_Services )
```

```
( initiator ignore )
```

```
( target ignore )
```

```
( outcome
```

```
( immediate
```

```
30    ( default Tcp_Access_Violation )
```

```
)
```

```
)  
)  
  
( rule Tcp_Connection_Terminated  
5      ( protocol TCP )  
      ( action ( union ABORT CLOSE TIMEOUT ) )  
      ( initiator ignore )  
      ( target ignore )  
      ( outcome  
10      ( immediate  
        ( default ok )  
      )  
    )  
  )  
)  
15  
  
( rule Tcp_Deny  
      ( protocol TCP )  
      ( action ignore )  
      ( initiator ignore )  
20      ( target ignore )  
      ( outcome  
      ( final  
        ( default Tcp_Access_Violation )  
      )  
25      )  
    )  
  )  
  
( rule Tcp_Missed_Connections  
      ( protocol TCP )  
30      ( action MISSED_CONNECT )  
      ( initiator ignore )
```

```
( target ignore )
( outcome
  ( immediate
    ( default Warn_Missed_Tcp_Connect )
5      )
  )
)

( rule Tcp_X_Dns_From_Colloc_To_Dns_Server
10   ( protocol TCP )
      ( action CONNECT )
      ( agent X_Monitors )
      ( initiator X_Coloc_Subnet )
      ( target Tcp_X_Dns_From_Colloc_To_Dns_Server_Target )
15   ( outcome
        ( immediate
          ( default ok )
        )
      )
20   )

( rule Tcp_X_Http_From_Any_To_All_Web_Servers_Provisional
      ( protocol TCP )
      ( action CONNECT )
25   ( agent X_Monitors )
      ( initiator ignore )
      ( target Tcp_X_Http_From_Any_To_All_Web_Servers_Provisional_Target )
      ( outcome
        ( final
30       ( default ok )
        )
      )
    )
```

```
)  
)  
  
( rule Tcp_X_Port_1984_Traffic  
5      ( protocol TCP )  
      ( action CONNECT )  
      ( agent X_Monitors )  
      ( initiator X_Coloc_Subnet )  
      ( target Tcp_X_Port_1984_Traffic_Target )  
10     ( outcome  
      ( immediate  
        ( default ok )  
      )  
    )  
15 )
```

```
( rule Tcp_X_X_Loghost_Traffic  
      ( protocol TCP )  
      ( action CONNECT )  
20     ( agent X_Monitors )  
      ( initiator X_Web_Servers )  
      ( target Tcp_X_X_Loghost_Traffic_Target )  
      ( outcome  
      ( immediate  
25        ( default ok )  
      )  
    )  
30 )
```

```
( rule Tcp_X_Shsh_From_Clouds_To_Cgi_Provisional  
      ( protocol TCP )
```

```
( action CONNECT )
( agent X_Monitors )
( initiator Clouds )
( target Tcp_X_ShH_From_Clouds_To_Cgi_Provisional_Target )
5      ( outcome
        ( final
          ( default ok )
        )
      )
10    )

( rule Tcp_X_Spm_Colloc_Traffic
      ( protocol TCP )
      ( action CONNECT )
15    ( agent X_Monitors )
      ( initiator Modin )
      ( target Tcp_X_Spm_Colloc_Traffic_Target )
      ( outcome
        ( immediate
20      ( default ok )
        )
      )
    )

25  ( rule Tcp_X_Spm_Colloc_Traffic_Provisional
      ( protocol TCP )
      ( action CONNECT )
      ( agent X_Monitors )
      ( initiator Modin )
30    ( target Tcp_X_Spm_Colloc_Traffic_Provisional_Target )
      ( outcome
```

```

        ( final
          ( default ok )
        )
      )
5      )

      ( rule Tcp_X_Ssh_From_Fluffy_To_Monkey_Provisional
        ( protocol TCP )
        ( action CONNECT )
10      ( agent X_Monitors )
        ( initiator Fluffy )
        ( target Tcp_X_Ssh_From_Fluffy_To_Monkey_Provisional_Target )
        ( outcome
          ( final
15          ( default ok )
          )
        )
      )

20      ( rule Tcp_X_Ssh_From_Monkey_To_Fluffy_Provisional
        ( protocol TCP )
        ( action CONNECT )
        ( agent X_Monitors )
        ( initiator Monkey )
25      ( target Tcp_X_Ssh_From_Monkey_To_Fluffy_Provisional_Target )
        ( outcome
          ( final
            ( default ok )
          )
30      )
    )
  )
```

```
( rule Tcp_X_Ssh_From_X_To_X_Web_Servers_Provisional
  ( protocol TCP )
  ( action CONNECT )
5  ( agent X_Monitors )
  ( initiator X_Ssh_From_X_To_X_Web_Servers_Provisional_Initiator )
  ( target Tcp_X_Ssh_From_X_To_X_Web_Servers_Provisional_Target )
  ( outcome
    ( final
10      ( default ok )
    )
  )
)

15 ( rule Tcp_X_Ssh_To_Security_Web_Server
  ( protocol TCP )
  ( action CONNECT )
  ( agent X_Monitors )
  ( initiator X_Ssh_To_Security_Web_Server_Initiator )
20 ( target Tcp_X_Ssh_To_Security_Web_Server_Target )
  ( outcome
    ( immediate
      ( default ok )
    )
25 )
)

( rule Tcp_X_Stmp_From_All_To_X
  ( protocol TCP )
30 ( action CONNECT )
  ( agent X_Monitors )
```

```
( initiator X_Stmp_From_All_To_X_Initiator )
( target _Smtip )
( outcome
  ( immediate
    ( default ok )
  )
)
)
)

10 ( rule Tcp_Unknown_Protocol
    ( protocol TCP )
    ( action PROTOCOL_UNKNOWN )
    ( initiator ignore )
    ( target ignore )
15 ( outcome
    ( immediate
      ( default Incorrect_Port_Usage )
    )
  )
20 )

( rule Udp_Deny
  ( protocol UDP )
  ( action ignore )
25 ( initiator ignore )
  ( target ignore )
  ( outcome
    ( immediate
      ( default Udp_Access_Violation )
30 )
  )
)
```

```

)

( rule Udp_X_Dns_From_Colloc_To_Dns_Server
  ( protocol UDP )
5   ( action ASSOCIATION )
    ( agent X_Monitors )
    ( initiator X_Colloc_Subnet )
    ( target Udp_X_Dns_From_Colloc_To_Dns_Server_Target )
    ( outcome
10   ( immediate
      ( default ok )
    )
  )
)
15 )

```

Table P

Evaluation Algorithm

In the preferred embodiment the policy engine applies a policy
 20 evaluation algorithm to each incoming protocol event. The algorithm
 results in a selection of a policy rule applicable to the protocol event
 and may produce an immediate or final disposition.

Following is a step-by-step description of the evaluation algorithm
 25 according to the preferred embodiment. It is noted that the evaluation
 procedure described herein below is in conceptual form and does not
 take into account any possible runtime optimizations:

- 1) Select a set of rules applicable to an Agent reporting an event;
- 2) From said set, select a second set of rules applicable to an associated examined protocol.

5

- 3) From said second set, select a third set of rules applicable to an associated examined protocol action.

10

- 4) Starting with a most specific policy rule in said third set and descending to a least specific rule find a policy rule satisfied by said protocol event. A matching algorithm according to the preferred embodiment is as follows:

15

- a) If one or more orderly listed prerequisite rules are specified, ensure at least one of said prerequisite rules is satisfied by a previously processed protocol event. In the preferred embodiment a prerequisite rule is satisfied if it is a pending policy rule for the protocol event.

20

- b) Match initiator and target credentials in the policy rule against the corresponding initiator and target credentials presented in the protocol event.

25

- 5) If a policy rule satisfying the protocol event is not found the policy engine generates a disposition for the network event indicating that

a policy specification error was encountered. Effectively the processing of the network event thereby terminates.

6) If a policy rule satisfying the protocol event is found, the policy engine checks for other rules having a same ranking number and also satisfying the event. If such rules are found the policy engine uses the following algorithm in the preferred embodiment to select a single applicable rule:

a) Rules that specify all protocols, *i.e.* using ignore or present, are less specific than rules that explicitly list a set of one or more protocols.

b) Rules that specify all actions (*i.e.* using ignore or present) are less specific than rules that explicitly list a set of one or more actions.

c) Rules that have prerequisites are more specific than rules that do not have prerequisites. Rules that specify a higher-ranking prerequisite are more specific than rules that specify a lower-ranking prerequisite. In the preferred embodiment a ranking relationship is relevant only if both prerequisite rules belong to a same protocol-action group.

d) If thereafter a single rule is determined as more specific than the others it is selected for the protocol event. If more than one rule

remains the policy engine sorts the remaining rules in increasing lexical order by name and selects a first rule from the sorted rules having an immediate disposition indicating in decreasing order of precedence:

5

- i) a policy violation (any disposition code other than OK or CONTINUE);
- ii) CONTINUE (allows other rules to examine further the network event); and
- 10 iii) OK

15

The outcome of the policy evaluation algorithm herein above is a policy rule that satisfies the protocol event. If an immediate outcome is specified for that rule, it is executed, producing a disposition for the protocol event. If the disposition comprises a final disposition code (any code other than CONTINUE), the disposition is also the final disposition for the network event.

20

Otherwise in the preferred embodiment the selected policy rule is a pending policy rule for the network event. In absence of any further protocol events the pending policy rule is promoted to selected policy rule. A final outcome of the selected policy rule is executed producing a final disposition for the network event.

25

An Exemplary User Interface for Providing and Reporting Processed and Analyzed Network Data to an End User

An exemplary user interface for providing and reporting the processed and analyzed network data from the database (Fig. 1a - 165) to an end user is provided below.

- 5 It should be appreciated that examples of a typical end user using such interface are, but are not limited to a customer whose network is being monitored, an operations analyst reviewing the customer's network environment and network data, and/or a policy analyst reviewing the network data and its conformance to network policy.

10

The preferred embodiment of the invention uses a web page paradigm as an example of a type of user interface, and is described with reference to figures of screen prints of web pages herein. While the claimed invention herein has disclosed a web page implementation of a user interface, it will be appreciated
15 by those skilled in the art that such user interface readily encompasses any form, that can be substituted therefore to effect a similar result as is achieved by the web page, including but not limited to any graphical user interface or non-graphical user interface.

- 20 The preferred embodiment of the invention is described with reference to Fig. 20 and comprises a system dashboard, label 20000 on a home page, wherein the dashboard 20000 is kept up to date with current monitoring information from the monitored network.

- 25 In the preferred embodiment of the invention, the dashboard 20000 updates once every five minutes. It should be appreciated that different update rates

can be used to keep the data on the dashboard 20000 current, and that parts of the underlying customer data may be updated at a different, such as a slower rate.

- 5 The preferred embodiment of the invention provides a tear off feature on the system dashboard 20000. In this example, the end user clicks on a tear off tab 20010 to open a tear off console window. Fig. 21 shows an example of a tear off console window according to the invention. It is intended that the end user keep the console window open on the computer desktop all day long to
10 view high level reporting of the health of the monitored network.

The preferred embodiment of the invention provides an outstanding alerts area 20020 of the dashboard and consists of a FIFO queue of CRITICAL alerts that have been generated by the policy monitoring system (Fig. 1a -
15 106). In the preferred embodiment of the invention the following applies. The size of the alert list can be limited to a predetermined number of elements. The total number of open alerts can be displayed within the alerts area 20030.

The underlying data is updated on a real-time basis. Entries in the list link to
20 alert details, as depicted in Fig. 28. In this example, clicking on an entry in the list 20030 opens up an alert details page 2801 for that particular alert, comprising such alert details as, for example rule, disposition, time of alert, type of alert, source ip-address, destination IP-address, and the like.

- 25 The preferred embodiment of the invention provides a health monitor 20040 to show a visual representation of the severity categories into which the current

observed traffic has been assigned over a predetermined amount of time. In this example, the underlying data is updated every five minutes and summarizes traffic over the last one hour and last twenty four hour periods.

CRITICAL and HIGH severity alerts have a red bar 20050, MEDIUM,
5 WARNING and MONITOR uses a yellow bar 20060, and all others are green 20070.

The preferred embodiment of the invention provides access to current summary reports. An example is shown in Fig. 20 as part of the end user's
10 home page. Such screen allows the end user to generate queries that summarize report data filtered by the monitoring point and over configurable time periods. An interface feature, such as a dropdown listbox 20090 allows the end user to choose one of a predetermined set of time periods, such as but not limited to the following:

15

- Select date range – A specific time period expressed in starting month, day and hour, followed by ending month, day and hour using an interface feature such as dropdown listboxes 20091;

20

- Last two hours;

- Last 24 hours;

- Today (since midnight);

25

- Yesterday (00:00 – 23:59:59);

- Last seven days;
- This month (from first to present);

5

- Last month (from first to end of month);
- Last three months (three months back from present); and

- 10 ◦ Custom (retrieves date/time range from the last manually configured query).

The preferred embodiment of the invention provides an events summary view as shown in Fig. 22.

15

In the example shown in Fig. 22, viewing the summary for a specific time period displays both a chart 2201 of a predetermined number of columns and a table 2202 displaying the following information, when the conformance tab 2203, the violators tab 2204, or the targets tab 2205, respectively, is selected:

20

- A conformance chart/table shown in Fig. 22, displaying the count of violations for each rule/disposition pair.

- An icon 2206 links to a network event details page, such as shown in Fig. 23 that contains details of events that make up this count, *i.e.* all

25

network events with such rule/disposition pair that occurred in the given time period.

- A violators chart 2901 and table 2902 shown in Fig. 29, displaying the count 2903 of the number of violations for each of the top violating ip-addresses 2904.

- An icon 2206 links to a network event details page, such as shown in Fig 23 that contains details of events that make up this count, *i.e.* all network events with such originating ip-address that occurred in the given time period.

- A targets chart 3001 and table 3002 shown in Fig. 30, displaying the count 3003 of the number of violations for each of the top destination IP-addresses 3004.

- An icon 2206 links to the a event details page, such as shown in Fig 23 that contains details of events that make up this count, *i.e.* all network events with such destination IP-address and port that occurred in the given time period.

Fig. 22 shows the events summary report for conformance.

The preferred embodiment of the invention provides a link to network events detail information. In this example, a separate link 2206 builds a network events details page as shown in Fig. 23. Fig. 23 contains a table that may be

sorted or reverse sorted by any of the columns displayed 2301 of all violating network events with such a rule/disposition pair that occurred in the chosen time period.

- 5 In the preferred embodiment of the invention, the summary page (Fig. 22) contains a specification of the date range of the data being displayed. In particular, if the start of the range falls outside the range of date for acquiring user data then the actual start date of the user data is displayed.
- 10 It should be appreciated that in another equally preferred embodiment, user defined and configurable query and reports settings can be stored, for example, in a user's preferences or profile.

The preferred embodiment of the invention comprises trend reports on the
15 dashboard, wherein such reports comprise charts that link to a network events summary page containing details of the summarized traffic. More specifically, the charts, unless otherwise explicitly specified, are bar charts, each of which link to the network events summary page.

- 20 Referring to Fig. 20, the preferred embodiment of the invention comprises a section, such as a QuickWeek section 20100 of the end user's main page, such as a login page or home page that contains trend graphs, such as but not limited to the following:

- 25 ◦ During the past seven days, the five most frequent rule/disposition combinations versus count 20110;

- During the past seven days, the five most frequent violator ip-addresses versus count 20120; and
- 5 ◦ During the past seven days, the five most frequent target ip-addresses versus count 20130.

It should be appreciated that another equally preferred embodiment of the invention comprises an input means for the end user to customize which
10 trends appear in the trend, *e.g.* QuickWeek section, and to customize the time period being viewed.

The preferred embodiment of the invention comprises trend charts that are embedded into details pages. Each of the trend charts allows the end user to
15 dynamically configure a time range by a means such as a pull down menu. Examples of such embedded trend charts are:

- Policy effectiveness;
- 20 ◦ Number of policy changes over time;
- Event Summary (such as for the following):
 - Conformance: Graphical view of the data for the specified time
25 period 2201;

- Violators: Graphical view of the data for the specified time period; and
 - Targets: Graphical view of the data for the specified time period; and
- 5 ▪ Network Event Details (such as for the following):
- Conformance Event Details (Fig. 23): Violator count over time for a particular rule/disposition combination 2303;
- 10 ◦ Violators Event Details: Conformance count over time for a particular violator; and
- Target Event Details: Conformance count over time for a particular target;
- 15 ◦ All, e.g. in chronological order: Conformance count over time for a particular time period.

The preferred embodiment of the invention provides event detail reports, such as for but not limited to network event details, protocol event details, and alert details, described below.

20

The preferred embodiment of the invention provides a network event details page containing listed fields in columns that vary according to the violation type, such as, for example, All, Conformance (Fig. 23), Violator, and Target that had been selected at the summary level. For each type, except All,

25

rather than repeat the field or column(s) which reiterate the violation, it will be displayed in the heading of the events detail page. For example, after choosing to view event details for a particular target, the DstIP is not repeated in every row. Each of the columns may be used to sort or reverse sort the report by clicking on that column's heading name. Following is a list of types of data provided in a network event details page:

- Monitoring Point;
- 10 ◦ Disposition Name;
- Rule Name;
- Disposition Code;
- 15 ◦ Severity;
- Src IP;
- 20 ◦ Src Port;
- Dst IP;
- Dst Port;
- 25 ◦ IPProtocol;

- Event Time: event times can be stored throughout the system in UTC; and

- Application Data:

5

- ICMP – ICMP action code;

- HTTP – URL;

10

- FTP – Filename;

- SSL – Ciphersuite, Issuer and Subject's certificate CommonName, Certificate Status;

15

- SSH – Authentication handshake status; and

- Application Status Code

- HTTP – StatusCode.

20

The preferred embodiment of the invention provides a protocol event details page as depicted in Fig. 24 and that is created in the context of a particular network event instance. This data is retrieved on an as-needed basis from a database. The content of this page reflects the data available in a protocol event view of the QueryTool and is specific to the protocol or protocols being displayed. Such data includes, but is not limited to:

25

- Data from such attributes as IP address, interface address, protocol ID, service port, URL, file pathname, user name, password metrics, public key certificate, encrypted session parameters and status codes; and

5

- Protocol-specific actions such as HTTP methods, TCP protocol messages, ICMP message codes, FTP control commands, and authentication steps.

The preferred embodiment of the invention provides an alert event details

10 page as depicted in Fig. 28 containing, but not limited to the following:

- details of the network event that caused the alert;
- rule and disposition name that triggered alert;

15

- log comment from the disposition;
- time at which the alert was generated;

20

- initiator ip address of the corresponding non-conformant traffic;
- target ip address of the corresponding non-conformant traffic;

25

- an icon that links to the network event details page describing the non-conformant network event; and

- checkbox to clear the alert.

The preferred embodiment of the invention provides a policy update page containing, but not limited to a table displaying each time a new policy is
5 installed on the security policy management system discussed herein. This table contains, but is not limited to:

- Date of the policy installation;
- 10 ◦ Description of policy; and
- A link to the English description that represents the newly installed policy.

It should be appreciated that in the preferred embodiment of the invention
15 alerts are generated whenever a disposition with a CRITICAL severity is assigned to a network event, each alert generating an email containing, but not limited to the following information:

- time the alert occurred;
- 20 ◦ rule and disposition name that triggered alert;
- log description, if any, from the corresponding disposition;
- 25 ◦ initiator ip address of the corresponding non-conformant traffic;

- target ip address of the corresponding non-conformant traffic; and
- link to the network event detail describing the non-conformant network event.

5

The preferred embodiment of the invention provides a customer page that allows the user to configure a list of email addresses within a customer's organization that shall receive alert email.

- 10 Another equally preferred embodiment provides means for accessing ad-hoc queries for the end user, such as, but not limited to, filtering results by any one or all of the following:

- Protocol of the rule name;

15

- Policy rule name;

- A regular expression within the rule name;

20

- Disposition name of the violation;

- A regular expression within the disposition name;

- Source ip-address;

25

- A regular expression with source ip-address;

- Target (Destination) ip-address;
- A regular expression within target (destination) ip-address;

5

- Target (destination) port; and
- A regular expression within target (destination) port.

10 An example of a means for accessing ad-hoc queries is an advanced search feature, such as for example, an advanced search dialog box 3100, as depicted in Fig. 31. In the preferred embodiment of the invention, the advanced search dialog box 3100 comprises list boxes for such categories, such as protocol 3101, rule 3102, and disposition 3103, and text boxes for
15 descriptions, such as regular expression in a rule 3104 or disposition 3105 and IP-addresses 3106.

In the preferred embodiment of the invention, an end user can open the advanced search dialog box 3100 from an Advanced Search link 3201 on the
20 dashboard, as depicted in Fig. 32, or from any event summary or event details page.

The preferred embodiment of the invention provides informational aids. For example, the following information about a user's policy is available via a
25 variety of features, such as but not limited to links, tool tips, and the like:

- Customer specific policy interpretation, such as provided by English language representation;
- 5 ◦ Rule and disposition descriptions as defined by the user in the user's policy, resolved DNS names for ip-addresses, and TCP and UDP service names; and
- A copyright page containing copyrights and trademarks as required by
- 10 licensing agreements with vendors.

The preferred embodiment provides links to descriptions of rules, dispositions, IP-addresses, and the like, displayed, for example in a pop up window whenever the user's cursor is over the respective field, as depicted in Fig. 22

15 2207, Fig. 23-2302, Fig. 25-2501, Fig. 26-2601, and Fig. 27-2701, respectively.

The preferred embodiment of the invention provides links on each page that include, but are not limited to:

20

- Context sensitive help per-page.

In the preferred embodiment of the invention, each details page contains a button linking to a printer friendly version of the page.

25

In the preferred embodiment of the invention, regardless of the time zone the user's or the policy monitoring systems runs on, such as, for example Universal Time Coordinates (UTC). Any time being displayed to the user, such as, for example, on a website or in contents of emails, is converted to
5 the user's time zone and as such is explicitly displayed.

A Declarative Language for Specifying A Security Policy

Herein below an equally preferred embodiment of the invention is provided,
10 and is discussed with reference to Figures 33-37.

Overview

Fig. 33 is a schematic diagram showing the relationship of elements of the policy monitoring system 100, according to the preferred embodiment of the
15 invention. To effect a security policy decision, a policy manager module invokes a policy engine 102 with both a reference to a pre-defined security policy and a number of inputs it received from an Agent 129. These inputs describe a protocol event such as a TCP connection, an SSL session, or an HTTP GET. The end to end interaction between two communicating entities
20 comprises one or more protocol events and it is termed a network event 130. For example, the retrieval of a web page from a web server by a web browser is a network event that typically consists of an IP protocol event, a TCP protocol event and an HTTP protocol event.

25 In the preferred embodiment the policy engine 102 consults a policy information database, a Policy Store 104 to determine a policy rule that

applies to the network event 130. In the preferred embodiment the policy engine 102 collects input from the Agent 129 about each protocol event until it has enough information to consult the Policy Store 104. Once an applicable policy rule for the entire network event 130 has been found, the policy engine

5 102 returns a disposition 136 for the event to the policy manager module which in turn forwards it to the Agent 129, to a logging subsystem and, optionally, to an enforcement subsystem.

A definition of a protocol event is provided to facilitate understanding of the

10 invention. A protocol event 120 as shown in comprises the following elements:

1) The Principals. Every policy decision involves two principals: an initiator 121 (an active principal) and a target 122 (a passive principal). Principals are

15 identified by a set of one or more credentials, depending on how much information is known about them and what protocol service they are using.

There are three types of credentials:

- a) Host credentials. These are the physical network address, *i.e.* a MAC address, and the network attachment point, *e.g.* an IP address and port
- 20 number.
- b) User credentials. These may be weak credentials, *e.g.* a user name and password, or strong credentials, *e.g.* an X.509 certificate.
- c) Object credentials. These identify a resource or an application, *e.g.* a
- 25 URL or a pathname.

2) The Protocol. The protocol service 123 associated with this protocol event 120.

3) The Security Quality of Service Parameters. Some protocols include security QOS parameters 124 and these may be subject to local security policy constraints. For example, in the SSL protocol the ciphersuite negotiated between the SSL client and the SSL server is a security QOS parameter.

4) The Action. Every interaction between an initiator and a target over a given protocol service involves a specific action 119. Clearly, not all actions are of interest to the policy manager module. For example, in the SSL protocol only actions pertaining to the establishment or termination of an SSL session, most notably, the negotiation of security parameters for the session are of interest. In the LDAP protocol, on the other hand, a security policy administrator may wish to express policy statements about different LDAP data manipulation operations, such as, the SEARCH and MODIFY operations.

In one embodiment of the invention, while processing a network event 130, and before issuing a final ruling, the policy engine 102 may instruct the Agent 129 to carry out specific actions against the network event 130. For example, the Agent 129 may be asked to decrypt subsequent SSL traffic or it may be asked to impose a specific ciphersuite on the target system. These instructions constitute an intermediate output of the policy engine 102 and are issued in the form of agent directives, defined herein below.

Once the policy engine 102 arrives at a final policy decision, it produces a disposition 136 for the event 130. The disposition 136 as shown in Fig. 35 comprises the following elements:

- 5 1) Disposition Code. The disposition code 131 denotes whether or not the event 130 complies with the security policy and, if not, identifies the specific policy violation. A list of possible codes in a preferred embodiment is given in Table S herein below. This field is mandatory.
- 10 2) Logging Directives. The logging directives field 132 includes a severity code, denoting the severity of the policy violation. A list of possible severity values in a preferred embodiment is given herein below in Table T. The severity code may be used by a logging subsystem to filter the event 130 and its disposition 136 or to control a notification action, *e.g.* page a network
15 operator. In another embodiment the logging directives 132 may also include an optional human readable description summarizing the specific policy that determined the final disposition 136 *e.g.* "blue users cannot access the red server". The logging directives field 132 is mandatory if the disposition code 131 indicates a policy violation.
- 20 3) Agent Directives. Agent directives 129 are any instructions that need to be communicated to the Agent 129 and in another embodiment to more than one Agent. For example, the Agent 129 may be instructed to log all traffic associated with the event 130 or to disrupt communications between the
25 initiator 121 and the target 122. In some embodiments, an Agent 129 only supports monitoring functions, or only enforcement functions, or be limited in

its support of other types of functions. In a preferred embodiment, a policy manager is responsible for distributing a set of directives to appropriate Agents.

- 5 4) Event Constraints. Event constraints 134 are any constraints to be applied to the event 130. For example, in one embodiment these constraints are protocol-specific constraints such as the maximum lifetime of a TCP connection or the maximum lifetime of an SSL session. In another embodiment, these constraints are communicated to the Agent reporting the event or simply to a policy manager.

Table S

Built-in Objects

The following is a set of built-in language objects known to both the policy compiler and the policy engine.

First the built-in groups. It should be noted that, unlike user-defined groups, built-in groups cannot be extended in a policy specification.

```

20  // List of supported protocols
    ( group all-protocols  protocol_t
      ( union IP UDP ICMP TCP SSL HTTP )
      // NOTE: new protocols can be added as needed
    )
25
    // List of supported hash algorithms
    ( group hash-algorithms hash_alg_t
      ( union MD5 SHA1 )
    )
30
    // List of supported agent directives

```

```
( group agent-directives agent_directive_t
    ( union DECRYPT DISRUPT LOG_TRAFFIC )
)

5 // List of supported logging severity codes
( group severity-codes severity_t
    ( union CRITICAL HIGH MEDIUM WARNING MONITOR INFORMATION )
)

10 // List of supported disposition codes
( group disposition-codes code_t
    ( union OK CONTINUE ACCESS_DENIED AUTHENTICATION_VIOLATION
        SECURITY_ATTACK SECURITY_QOS POLICY_ERROR )
)

15 // Certificate status values for valid certificates
( group valid-certs cert_status_t
    ( union VALID )
)

20 // Certificate status values for certificates rendered invalid
( group invalid-certs cert_status_t
    ( union EXPIRED NOT_YET_VALID REVOKED SUSPENDED )
)

25 // Certificate status values for rejected certificates
( group rejected-certs cert_status_t
    ( union MALFORMED UNSUPPORTED_CRITICAL_EXTENSION )
)

30 // Certificate status values for all bad certificates
( group bad-certs cert_status_t
    ( union rejected-certs invalid-certs )
)

35 // List of all possible certificate status values
( group cert-status-values cert_status_t
    ( union valid-certs invalid-certs rejected-certs )
)

40
```

```

// List of all possible authentication status values
( group auth-status-values auth_status_t
  ( union SUCCEEDED REJECTED ABORTED )
)
5
// List of all SSL ciphersuites
( group ssl-ciphersuites ciphersuite_t
  ( union SSL_RSA_WITH_NULL_MD5
10      SSL_RSA_WITH_NULL_SHA
      SSL_RSA_EXPORT_WITH_RC4_40_MD5
      SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
      SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
      SSL_RSA_WITH_RC4_128_MD5
      SSL_RSA_WITH_RC4_128_SHA
15      SSL_RSA_WITH_IDEA_CBC_SHA
      SSL_RSA_WITH_DES_CBC_SHA
      SSL_RSA_WITH_3DES_EDE_CBC_SHA
      SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
      SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA
20      SSL_DH_RSA_WITH_DES_CBC_SHA
      SSL_DH_DSS_WITH_DES_CBC_SHA
      SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
      SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
      SSL_DH_ANON_EXPORT_WITH_RC4_40_MD5
25      SSL_DH_ANON_WITH_RC4_128_MD5
      SSL_DH_ANON_EXPORT_WITH_DES40_CBC_SHA
      SSL_DH_ANON_WITH_DES_CBC_SHA
      SSL_DH_ANON_WITH_3DES_EDE_CBC_SHA
      SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
30      SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
      SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
      SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
      SSL_DHE_RSA_WITH_DES_CBC_SHA
      SSL_DHE_DSS_WITH_DES_CBC_SHA
35      SSL_FORTEZZA_KEA_WITH_NULL_SHA
      SSL_FORTEZZA_KEA_WITH_FORTEZZA_CBC_SHA
      SSL_FORTEZZA_KEA_WITH_RC4_128_SHA
      SSL_V2_RC4_128_WITH_MD5
      SSL_V2_RC4_128_EXPORT40_WITH_MD5
40      SSL_V2_RC2_CBC_128_CBC_WITH_MD5

```

```
        SSL_V2_RC2_CBC_128_CBC_EXPORT40_WITH_MD5
        SSL_V2_IDEA_128_CBC_WITH_MD5
        SSL_V2_DES_64_CBC_WITH_MD5
        SSL_V2_DES_192_EDE3_CBC_WITH_MD5
5      )
    )

    // List of supported action codes for TCP
    ( group tcp-action-codes action_t
10      ( union CONNECT
          MISSED_CONNECT
          TIMEOUT
          ABORT
          CLOSE )
15    )

    // List of supported action codes for UDP
    ( group udp-action-codes action_t
20      ASSOCIATION
    )

    // List of supported action codes for IP
    ( group ip-action-codes action_t
25      ASSOCIATION
    )

    // List of supported action codes for ICMP
    ( group icmp-action-codes action_t
30      ( union ASSOCIATION
          BAD_CODE
          FRAGMENTATION_NEEDED
          HOST_UNREACHABLE
          NETWORK_UNREACHABLE
          PORT_UNREACHABLE
35      PROTOCOL_UNREACHABLE
          SOURCE_ROUTE_FAILED
          ECHO
          ECHO_REPLY
          INFORMATION_REQUEST
40      INFORMATION_REPLY
    )
    )
```

```

// List of supported action codes for SSL
( group ssl-action-codes action_t
15   ( union  HANDSHAKE
        MISSED_HANDSHAKE
        SESSION_CLOSED
        SESSION_ABORTED )
    )

20 // List of supported action codes for HTTP
( group http-action-codes action_t
    ( union  GET
        HEAD
25     POST
        PUT
        DELETE
        OPTIONS
        TRACE
30     CONNECT
        MISSED_REQUEST
        RESPONSE )
    )

35 // List of all supported action codes
( group all-action-codes action_t
    ( union  udp-action-codes
        ip-action-codes
        icmp-action-codes
40     tcp-action-codes
    )
)

```

```
        ssl-action-codes
        http-action-codes )
)
```

- 5 Now, the dispositions and policy rules built into the Policy Engine. These rules can be overwritten by user-defined policy rules.

```
( disposition ok
  ( code OK )
10 )

( disposition continue
  ( code CONTINUE )
15 )

( disposition policy-error
  ( description "Policy error caused by uncaught event". )
  ( code POLICY_ERROR )
  ( log-directive
20     CRITICAL
    "Uncaught event" )
  )

25 ( rule default-rule
    ( description "Catch-all rule for all protocols". )
    ( protocol present )
    ( action present )
    ( initiator ignore )
30    ( target ignore )
    ( outcome
      ( final
        ( default policy-error )
      )
    )
35 )
)
```

It is noted that the list of built-in objects included in Table S is by no means complete. In other embodiments, the set of built-in objects is expanded or reduced to reflect the set of protocols supported by the Policy Monitoring System.

5

It is noted that in the preferred embodiment the policy engine 102 ranks default-rule lower than any user-defined rule. For example, a user-defined rule having initiator and target credentials set to ignore ranks higher than using default-rule.

10

In a preferred embodiment, security policy decisions are also affected by any previous history of security violations involving one or both of the principals.

Fig. 36 is a schematic diagram of communicating parties according to the invention; wherein an initiator host machine 144 attempts to contact a target

15 host machine 145 over a network, and its events are listened to by an Agent

129 and events are passed onto the invention herein 100. For example, a host machine 144 that repeatedly attempts to perform an illegal operation within a given time window may be blacklisted and rendered incapable of conducting further communication activity within the security domain. In one

20 embodiment, a policy manager maintains a count of security policy violations perpetrated by or against specific principals and provides that information to the policy engine 102 as input to a policy evaluation procedure.

Specification Language

25 A security policy is formulated using the Policy manager module's policy specification language (Fig. 1) 108. A preferred embodiment chooses a

- simplified form of S-expressions as the policy specification language 108. S-expressions are LISP-like parenthesized expressions. The preferred embodiment uses a variant of S-expressions devised by Ron Rivest in R. Rivest, *code and description of S-expressions*,
5 <http://theory.lcs.mit.edu/~rivest/sexp.html>, and used in SPKI/SDSI in C. Ellison, *SPKI Certificate Documentation*,
<http://www.clark.net/pub/cme/html/spki.html>. In the preferred embodiment the use of Rivest's S-expressions are restricted in a way such that no empty lists are allowed and such that each list must have a type (a byte string) as its first
10 element, denoting the type of the object represented by the list. The use of Rivest's S-expressions is further restricted by only requiring support for the canonical and advanced representations of S-expressions, a preferred embodiment of which is depicted in Table T herein below.
- 15 An advantage of using the canonical representation of S-expressions in the preferred embodiment is for digital signature purposes as well as for relatively efficient communication. It is easy to parse, fairly compact, and is unique for any given S-expression. An advantage of using the advanced representation of S-expressions is for human consumption. It can be thought of as a pretty
20 print of the canonical representation.

Table T

- 25 An example of an advanced representation:

```
( certificate ( issuer alice ) ( subject bob ) )
```

An example of a canonical representation:

It should be noted that replacing language tokens (*e.g.* certificate, issuer) with minimally encoded identifiers further optimizes the canonical representation.

The main advantages of using S-expressions in the preferred embodiment are:

- It is easy to represent arbitrary data with S-expressions.
- S-expressions are easy to extend and modify.
- In their advanced representation they are easy to read and edit using, for example, a simple text editor.
- Their canonical representation was designed for efficient packing and parsing. In particular, parsing requires minimal look-ahead and no re-scanning.
- Their canonical representation allows for easy transportation, for example, in files or email messages.
- In their canonical encoding they can be digitally signed.
- It is relatively simple to convert between the advanced and the canonical representation of S-expressions.

A formal description of the policy specification language 108 is provided herein below in Table U.

This table contains a Backus-Naur Form (BNF) description of the grammar for the policy specification language, including an annotation section. All valid policies derive from the <policy> production.

This grammar applies to the policy specification after all comments are removed and all macros are expanded. Comments begin with “//” and extend to the end-of-line. Macros are defined using the C macro syntax.

Incomplete parts of the grammar are noted in *italics*. Terminals are shown in **bold**.

10

```
//
// Basic language stuff
//
```

15

```
// Terminals requiring further syntax specification
```

```
    <integer> ::= TBD           // [0-9]*
    <symbol> ::= TBD           // alphanumeric and '-',
                                // '_', starts with
```

```
letter
```

20

```
    <string> ::= <concat> | TBD // any ASCII character
                                // enclosed in double-
```

```
quotes
```

```
    <mac-addr> ::= TBD           // 6 hex byte values
                                // separated by '-'
```

25

```
    <ip-addr> ::= TBD           // IPv4 dotted decimal
                                // notation
```

```
    <ip-mask> ::= TBD           // address prefix per
                                // RFC-2280
```

```
    <hex-string> ::= TBD        // n hex byte values
```

30

```
    <version-string> ::= TBD     // separated by ':'
                                // a string of the form
                                // <major>.<minor>
```

```
// Some productions used only for clarity
```

35

```
    <name> ::= <symbol>
    <type> ::= <symbol>
```

```

    <attr-name> ::= <symbol>

// The basic types in the language
5      <atom> ::= <symbol> | <integer> | <string> |
           <ip-addr> | <mac-addr> |
           <version> | <hash-atom> | <bool>

//
10 // Productions for the policy specification section
//

// These are values that describe the values of things with values
    <meta-value> ::= present | absent | ignore
15

// Productions used in a few places
    <assertion> ::= ( assertion <bool-expr> )

// Version conversion function
20      <version> ::= ( version <version-string> )

// Attributes, used as arguments in predicates and other operations
    <attr-part-list> ::= <atom> |
           <attr-part-list> <atom>
25      <attr-op> ::= ( <attr-name> <attr-part-list> )
    <attribute> ::= <attr-name> | <attr-op>

// Hashes
    <hash-alg-name> ::= [Some set of terminals of type hash_alg_t]
30      <hash-op> ::= ( hash <hash-alg-name> <attribute> )
    <hash-atom> ::= <hex-string>
    <hash> ::= <hash-atom> | <hash-op>

// Operations that operate on attributes and return a
35 // basic type (atom)
    <atom-ops> ::= <hash-op>

// Predicates - used in building boolean expressions
    <generic-compare-op> ::= eq
40      <num-compare-op> ::= gt | lt | ge | le

```

```

    <rng-compare-op> ::= range
    <string-compare-op> ::= prefix | substring
    <member-compare-op> ::= member
    <ipmask-compare-op> ::= ip-mask
5    <iprng-compare-op> ::= ip-range
    <cred-match-op> ::= root | has
    <presence-op> ::= present | absent

// Generic argument
10    <arg> ::= <attribute> | <atom> | <atom-ops>

    <generic-cmp-arg> ::= <arg>
    <generic-compare> ::= ( <generic-compare-op>
                            <generic-cmp-arg> <generic-cmp-arg> )
15

    <num-cmp-arg> ::= <attribute> | <integer> | <version>
    <num-compare> ::= ( <num-compare-op>
                        <num-cmp-arg> <num-cmp-arg> ) |
                        ( <rng-compare-op>
20                        <num-cmp-arg>
                        <num-cmp-arg> <num-cmp-arg> )

    <string-cmp-arg> ::= <attribute> | <string>
    <string-compare> ::= ( <string-compare-op>
25                        <string-cmp-arg> <string-cmp-arg> )

    <member-cmp-arg> ::= <arg>
    <member-part-list> ::= <member-cmp-arg> <union> |
                        <member-cmp-arg> <member-cmp-arg>
30    <member-compare> ::= ( <member-compare-op> <member-part-list> )

    <ipmask-compare> ::= ( <ipmask-compare-op>
                            <attribute> <ip-mask> )
    <iprng-cmp-arg> ::= <attribute> | <ip-addr>
35    <iprng-compare> ::= ( <iprng-compare-op>
                            <iprng-cmp-arg>
                            <iprng-cmp-arg> <iprng-cmp-arg> )

    <cred-match> ::= ( <cred-match-op> <attribute> <cred-name>
40    )

```

```

    <presence> ::= ( <presence-op> <attribute> )

    <predicate> ::= <generic-compare> | <num-compare> |
5      <string-compare> | <member-compare> |
      <ipmask-compare> | <iprng-compare> |
      <cred-match> | <presence>

10 // Boolean expressions
    <bool-list-op> ::= and | or
    <bool-monadic-op> ::= not
      <bool> ::= TRUE | FALSE

15    <bool-list> ::= <bool-expr> | <bool-list> <bool-expr>
    <bool-expr> ::= <name> | <bool> | <predicate> |
      ( <bool-monadic-op> <bool-expr> ) |
      ( <bool-list-op> <bool-list> )

20 // Descriptions are comments that are carried along with the
    constructs
    <string-list> ::= <string> | <string-list> <string>
    <description> ::= ( description <string-list> )

25 // When we need to break up a string across lines, we use concat to
    // put it back together (just like descriptions)
    <concat> ::= ( concat <string-list> )

30 // Unions are unnamed collections of one or more symbols
    // (with matching types)
    <union-symbol> ::= <atom> | <group-name>
    <union-list> ::= <union-symbol>
35      <union-list> <union-symbol> |
      ( union <union-list> ) |
      <union-list> ( union <union-list> )
    <union> ::= ( union <union-list> ) | <union-symbol>

40

```

```

// Groups are named and typed unions (all symbols must have one type)
<group-part-list> ::= <union> | <description> <union> |
                    <union> <description>
5    <group-name> ::= <name>
    <group> ::= ( group <group-name> <type>
                <group-part-list> )

10 // Credentials
    <cred-part> ::= <description> | <assertion>
    <cred-part-list> ::= <cred-part> | <cred-part-list> <cred-part>
    <cred-name> ::= <name>
    <credential> ::= ( credential <cred-name>
15                        <cred-part-list> )

// Dispositions
    <agent-directives> ::= [Some set of terminals of type
20                        agent_directive_t]
    <agent-directive-list> ::= <agent-directives> |
                            <agent-directive-list> <agent-directives>
    <agent-directive> ::= ( agent-directive <agent-directive-list> )

25 // preliminary list of severity codes; more TBD
    <log-severity> ::= [Some set of terminals of type severity_t]
    <log-directive> ::= ( log-directive <log-severity> <string> )
    |
    ( log-directive <log-severity> )

30 // preliminary list of disposition codes; more TBD
    <disposition-code> ::= [Some set of terminals of type code_t]
    <disp-code> ::= ( code <disposition-code> )

35    <disp-part> ::= <description> | <disp-code> |
                    <log-directive> | <agent-directive>
    <disp-part-list> ::= <disp-part> | <disp-part-list> <disp-part>
    <disposition-name> ::= <name>
    <disposition> ::= ( disposition <disposition-name>
40                        <disp-part-list> )

```

```

// Conditions
<condition-part-list> ::= <description> <assertion> |
                        <assertion> <description> |
5                        <assertion>
    <condition-name> ::= <name>
    <condition> ::= ( condition <condition-name>
                      <condition-part-list> )

10
// Outcomes are bindings of conditions to dispositions used in rules
<outcome-default> ::= ( default <disposition-name> )
    <guard> ::= if | ifnot
    <outcome-clause> ::= ( <guard> <condition-name>
15                          <disposition-name> )
    <outcome-list> ::= <outcome-default> |
                      <outcome-clause> <outcome-list>
    <immediate> ::= ( immediate <outcome-list> )
    <final> ::= ( final <outcome-list> )
20 <outcome-types> ::= <immediate> | <final> |
                      <immediate> <final> |
                      <final> <immediate>
    <outcome> ::= ( outcome <outcome-types> )

25
// Rules
    <protocol> ::= ( protocol <union> ) |
                  ( protocol <meta-value> )
    <action> ::= ( action <union> ) |
30              ( action <meta-value> )

    <initiator> ::= ( initiator <cred-name> ) |
                  ( initiator <meta-value> )
    <target> ::= ( target <cred-name> ) |
35              ( target <meta-value> )
    <agent> ::= ( agent <cred-name> )
    <rule-name> ::= <name>
    <rule-list> ::= <rule-name> | <rule-list> <rule-name>
    <prerequisite> ::= ( prerequisite <rule-list> )
40 <rank-above> ::= ( rank-above <rule-name> )

```

```

    <rule-part> ::= <description> | <protocol> | <action> |
                  <initiator> | <target> | <outcome> |
                  <prerequisite> | <rank-above> | <agent>
5    <rule-part-list> ::= <rule-part> | <rule-part-list> <rule-part>
    <rule> ::= ( rule <rule-name>
                <rule-part-list> )

10 // Policy is the top-level construct of the policy
    // specification section
    <policy-part> ::= <description> | <group> | <credential> |
                    <condition> | <disposition> | <rule>
    <policy-part-list> ::= <policy-part> |
15    <policy-part-list> <policy-part>
    <policy-name> ::= <name>
    <language-version> ::= <version-string>
    <policy> ::= ( policy <policy-name> <language-version>
                  <policy-part-list> )

20

    //
    // Productions for the annotation section
    //

25    <assertion-type> ::= <meta-value> | single-value | multi-value
    <weight> ::= ( weight <symbol> <assertion-type> )
    <weight-penalty> ::= ( weight-penalty <integer> )
    <rank-cred-part> ::= <weight> | <weight-penalty>
30 <rank-cred-part-list> ::= <rank-cred-part> |
    <rank-cred-part-list> <rank-cred-part>
    <ranked-credential> ::= ( ranked-credential <cred-name>
                             <rank-cred-part-list> )

    // Annotation is the top-level construct of the annotation section
35 <annotation-part-list> ::= <ranked-credential> |
    <annotation-part-list> <ranked-credential>
    <annotation> ::= ( annotation <policy-name>
                        <language-version>
                        <annotation-part-list> )

```

In the preferred embodiment the policy specification language 108 is typed. The policy compiler 101 performs the necessary type checking of all S-expressions found in a policy specification 105. Typing aids in catching and diagnosing both common and subtle user errors. A preferred embodiment of

5 the type information is described herein below in Table V.

Table V

Types

The subtables in this section describe, in a pseudo-formal manner, the type

10 information in the language that is enforced by the parser. The types used should be self-evident.

First some notation used throughout the tables:

- 15 ◦ (**list-of** type) - (foo (list-of T)) → (foo A B C) where A, B, and C are of the type T; the list must have at least one element.
- 20 ◦ (**multi-of** type) - (foo (multi-of T)) → (foo (union A B C)) where A, B, and C are of the type T or → (foo ABC) where ABC is the name of a group of type T.
- 25 ◦ (**mix-of** type1 type2 type3) - (foo (mix-of R S T)) → (foo A B C D) where A, B, C, and D are randomly of types R, S, and T.
- 30 ◦ **match** - a type that is required to be the same as all other occurrences of match in the expression - (foo match match) requires that the two arguments of foo be of the same type (e.g., int_t, string_t).

The following table lists the typed attributes used in conditions and

30 credentials.

Attribute Name	Applicable Protocols	Argument Types	Result Type
agent-attribute	all-protocols	—	(multi-of agent_attr_t)
auth-status	SSL		auth_status_t
cert-status	SSL	—	cert_status_t
der-cert	SSL	—	octet_string_t
encipher-keysize	SSL	—	int_t
http-cookie	HTTP	string_t	(multi-of string_t)
http-password	HTTP	—	string_t
http-req-hdr	HTTP	string_t	string_t
http-resp-hdr	HTTP	string_t	string_t
http-set-cookie	HTTP	string_t	(multi-of string_t)
http-status-code	HTTP	—	int_t
http-username	HTTP	—	string_t
icmp-gateway-address	ICMP	—	ip_addr_t
icmp-nested-address	ICMP	—	ip_addr_t
icmp-nested-port	ICMP	—	int_t
initiator-access-rate	all-protocols	—	int_t
initiator-auth-keysize	SSL	—	int_t
initiator-violation-rate	all-protocols	—	int_t
ip-address	IP UDP TCP ICMP	—	ip_addr_t
ip-port	IP UDP TCP ICMP	—	int_t
ke-keysize	SSL	—	int_t
mac-address	IP UDP TCP ICMP	—	mac_addr_t
protocol-version	all-protocols	—	version_t
ssl-ciphersuite	SSL	—	ciphersuite_t
target-access-rate	all-protocols	—	int_t
target-auth-keysize	SSL	—	int_t
target-violation-rate	all-protocols	—	int_t
url	HTTP	—	string_t
x509-cert-path	SSL	—	cert_path_t
x509-issuer	SSL	—	string_t
x509-subject	SSL	—	string_t

The table below lists all the operations in the language that return a dynamic result. For each operation it shows both argument and result types

Operation	Result Type	Argument Types
absent	bool_t	string_t
and	bool_t	(list-of bool_t)
default	disposition_t	disposition_t
eq	bool_t	<i>match match</i>
has	bool_t	cert_path_t cred_t
hash	base64_t	hash_alg_t octet_string_t
ge ¹	bool_t	<i>match match</i>
gt ¹	bool_t	<i>match match</i>
if	disposition_t	condition_t disposition_t
ifnot	disposition_t	condition_t disposition_t
ip-mask	bool_t	ip_addr_t ip_mask_t
ip-range	bool_t	ip_addr_t ip_addr_t ip_addr_t
le ¹	bool_t	<i>match match</i>
lt ¹	bool_t	<i>match match</i>
member	bool_t	<i>match (multi-of match)</i>
not	bool_t	bool_t
or	bool_t	(list-of bool_t)
prefix	bool_t	string_t string_t
present	bool_t	string_t
range ¹	bool_t	<i>match match match</i>
root	bool_t	cert_path_t cred_t

¹ Operator only supports types int_t and version_t as arguments.

substring	bool_t	string_t string_t
version	version_t	string_t

The table below is pushing the concept of "type" far beyond its normal meaning since, in it, we often use type merely to convey positional information. It shows the type of every object in the language and the types of

5 their arguments.

Object Name	Object Type	Argument Types
action	act_t	(multi-of action_t)
agent	agt_t	credential_t
agent-directive	agtdir_t	(multi-of agent_directive_t)
assertion	assert_t	bool_t
code	code_def_t	code_t
condition	cond_t	condition_t (mix-of desc_t bool_t)
credential	cred_t	credential_t (mix-of assert_t desc_t prot_t)
description	desc_t	(list-of string_t)
disposition	disp_t	disposition_t (mix-of desc_t code_def_t log_t agtdir_t)
final	dispo_t	(list-of guard_t)
group	group_t	<i>match</i> type_t (multi-of <i>match</i>)
immediate	dispo_t	(list-of guard_t)
initiator	init_t	credential_t
log-directive	log_t	severity_t string_t
outcome	out_t	(list-of dispo_t)
policy	policy_def_t	policy_t string_t (mix-of desc_t group_t cred_t cond_t disp_t rule_def_t)
prerequisite	pre_t	(list-of rule_t)
protocol	prot_t	(multi-of protocol_t)
rank-above	rank_t	rule_t
rule	rule_def_t	rule_t (mix-of desc_t agt_t prot_t act_t init_t targ_t out_t pre_t rank_t)
target	targ_t	credential_t
union	(multi-of <i>match</i>)	(list-of <i>match</i>)

It is noted that the list of credential and condition attributes included in Table V is by no means complete. In other embodiments, the set of attributes is expanded or reduced to reflect the set of protocols supported by the Policy Monitoring System.

5

It is noted that although the remainder of this disclosure describes the specification language 108 by means of examples, and that for improved readability, said examples use the advanced rather than the canonical representation of S-expressions, this is not meant to further limit the invention.

10

In the preferred embodiment of the invention, the language 108 allows for comments to be embedded in S-expressions. A comment is allowed anywhere whitespace is valid. A comment begins with `"/"` and continues to the end-of-line. In compilation, comments are ignored because they serve
15 merely as an aid to the human user.

In the preferred embodiment of the invention, the language 108 allows for external files to be included using the `#include` syntax of C. Included files are supported to enhance modularity and reusability of policy language segments.

20

In the preferred embodiment of the invention, the language 108 allows for macros to be defined using the `#define` syntax of C. Macros are supported to enhance readability. By convention, macros start with an uppercase letter but need not be fully capitalized.

25

The language 108 comprises the following first-class objects:

- Condition
- Credential
- Disposition
- Group
- 5 ◦ Policy
- Rule

In the preferred embodiment first-class objects have names. Names are normally used to refer to an object from another object. By convention,
10 names of built-in objects start with a lowercase letter and use hyphens (-) to separate words. Names of user-defined objects start with an uppercase letter and use intercaps or underscores (_) to separate words, but do not use hyphens. Names of data types start with a lowercase letter and end with an underscore followed by a lowercase 't' (_t).

15

In the preferred embodiment a named object must be defined before its name can be used. The scope of a name is that of the entire policy specification as defined by the policy object.

20 In the preferred embodiment first-class objects may optionally include a description field. The description provides human readable text associated with the object. Unlike comments, description fields are preserved by the policy parser. When using the advanced representation, description strings may be split across several lines, using the C rules of string concatenation.
25 That is, following the description token are one or more character strings, each enclosed in a set of double quotes.

Policy

In the preferred embodiment a policy is the top-most object defined by the specification language 108 and includes all other first-class objects. A policy manager may load several policies into its internal database. However, at any one point in time, only one active policy is in effect. That is the policy known to the policy engine 102. Following is an example of a policy object.

```
( policy Sample_Policy_1 "1.0"          // policy <name> <version>
  ( description "This is a policy specification description"
    "that is continued on a second line" )
  ...
)
```

In the preferred embodiment a policy object has two mandatory parameters: name, which is used to reference the policy, and version number, which defines the version of the policy specification language 108. A policy's version number is used to check for compatibility between a policy specification and a policy compiler.

Groups and Unions

In the preferred embodiment groups are named collections of a given type. The union object creates the collection from a set of items. The group object gives the union a name and a type. Following is an example expressing a collection of colors:

```
( group SomeColors color_t          // group <name> <type>
  ( description "Some colors I like" )
  ( union RED GREEN YELLOW )
)
```

In the example, the object identifies RED, GREEN and YELLOW as items, *i.e.* symbols, of type `color_t` (a fictitious data type) collected in a set named `SomeColors`. By convention, symbols defined in unions are fully capitalized.

5

In the preferred embodiment once a symbol is identified as being of a certain type, it is transparently added to an unnamed set of items of that type. It may then be reused in other unions, groups or wherever an individual item of that type is valid. For example, a valid way to define another group is as follows:

10

```
( group RedByAnyOtherName color_t
  ( description "Red in different languages" )
  ( union RED ROSSO ROUGE VERMELHO )
)
```

15

However in the preferred embodiment the following group would not be allowed since RED would already have been tagged as being of type `color_t`.

20

```
( group AfewOfMyFavoriteThings thing_t
  ( union RED PASTA WINE )    // ERROR! RED previously defined as
)                             // having type color_t
```

In the preferred embodiment sets can be combined with other predefined sets. For example,

25

```
( group MoreColors color_t
  ( union
    SomeColors
    RedByAnyOtherName      // overlapping ok
    PURPLE BEIGE BURGUNDY
```

30

)

It is noted that RED overlaps both SomeColors and RedByAnyOtherName,
5 which according to the invention is perfectly acceptable. The resulting set will
include only one instance of the set item RED.

In the preferred embodiment unions are similar to the C enum type, with the
added benefit that unions can be combined and extended without concern for
10 conflicting item values.

In a preferred embodiment unions are used, but are not limited to, to define
collections of items, such as, for example, IP addresses, MAC addresses,
integers, version numbers and hash values. That is, unions can define any
15 data item that has a primitive data type in the language. An example of a
group of IP addresses is defined as:

```
( group MyComputers ip_addr_t
  ( union
20      207.5.63.23          // desktop at work
      207.5.63.42          // laptop
      128.7.16.64          // home computer
  )
)
```

25

In the preferred embodiment the type of the items in the union must agree
with the type specified in the group.

In a preferred embodiment, groups are referenced from other first-class objects. For example, groups are typically used to define collections of protocol actions, SSL ciphersuites, and IP addresses. Note that wherever a group is allowed, the following are also valid:

- 5 ◦ A union object (essentially, an unnamed group) provided that any symbols used as elements in the union have already been given a type via a group definition.
- A single collection item. This is equivalent to a union object with a single element. If the item is a symbol, its type must have been previously
- 10 defined in a group.

A list of built-in groups is given in section Table S.

Credentials

- 15 In the preferred embodiment a credential is a statement about a principal in a protocol event. It consists of a logical expression containing one or more assertions about the attributes that make up a principal's credentials. When a policy rule is evaluated against a protocol event, the credentials presented in the protocol event are compared to the credentials specified in a purported
- 20 credential object. If the logical expression defined in the credential object is satisfied, the principal's presented credentials are said to satisfy the purported credentials. As an example, the following purported credentials are satisfied if the principal's IP address is 207.5.63.8 and its IP port number is either 80 or greater than 443.

25

```
( credential Credentials_Example_1           // credential <name>
```

```

    ( assertion
      ( and
        ( eq ip-address 207.5.63.8 )
        ( or
5          ( eq ip-port 80 )
          ( gt ip-port 443 )
        )
      )
    )
10 )

```

In the preferred embodiment each protocol has a set of attributes that may be used to build purported credentials. Table W herein below lists all the attributes currently defined and, for each attribute, it shows the protocols where the attribute might be included in the presented credentials, as well as

15 the operations where the attribute may be used as an operand.

Table W

Attribute Name	Applicable Protocols	Description	Compare Operations
<i>agent-attribute</i>	all-protocols ²	The attributes of the reporting Agent, as a union of symbolic names	member
<i>cert-status</i>	SSL	The validity status of a certificate	eq, member
<i>der-cert</i>	SSL	A DER encoded certificate	hash
<i>http-password</i>	HTTP	The password used in basic authentication	eq, member, substring, prefix
<i>http-username</i>	HTTP	The user name used in basic authentication	eq, member, substring, prefix
<i>ip-address</i>	IP UDP TCP ICMP	An IP address	eq, member, ip-mask, ip-range
<i>ip-port</i>	IP UDP TCP ICMP	An IP port	eq, member, gt, ge, lt, le,

² Can be used to identify the reporting Agent in any policy rule but must not be mixed with other credential attributes.

Attribute Name	Applicable Protocols	Description	Compare Operations
			range
<i>mac-address</i>	IP UDP TCP ICMP	A MAC address	eq, member
<i>url</i>	HTTP	A URL	eq, member, substring, prefix
<i>x509-cert-path</i>	SSL	An X.509 certificate chain	root, has
<i>x509-issuer</i>	SSL	An X.509 certificate issuer	eq, member, substring, prefix
<i>x509-subject</i>	SSL	An X.509 certificate subject	eq, member, substring, prefix

It is noted that the list of credential attributes included in Table W is by no means complete. In other embodiments, the set of attributes is expanded or reduced to reflect the set of protocols supported by the Policy Monitoring System.

In the preferred embodiment each attribute can be thought of as having an implied getter function that returns its value. Most attribute getters take no arguments and return a single value. In the preferred embodiment, however, some attribute getters (e.g. `http-req-hdr` and `http-cookie`) are functions that take one or more arguments and may return complex results. For example, `http-cookie` takes as an argument the name of a cookie in an HTTP request header and returns its value or values as a union of strings.

In the preferred embodiment it is important not to mix credential attributes from different protocol sets in a credential specification. For example, combining `ip-address` and `der-cert` in the same credential object would

be an error and flagged by the policy compiler. As another example, using a credential in a policy rule for a protocol action that is incompatible with the credential attributes in the credential object is considered an error, flagged by the policy compiler. However, it is possible to use those attributes in two
5 separate credential objects and establish relationships between them within policy rules (*e.g.* access to resource X is restricted to principals previously authenticated with credentials Y). See example Check_Access_Denial herein below for an example of establishing this type of relationships in policy rules.

10 In the preferred embodiment the credential attribute agent-attribute is used to define the credentials of the Agent 129 reporting the protocol event 130. Agents are individually configured with a set of attributes, which are used to identify them to a policy manager. In another embodiment, some agent attributes might uniquely identify a specific Agent (*e.g.*
15 MONITOR_NEXT_TO_ROUTER_X) while others might identify a group of Agents (*e.g.* ALL_MONITORS_IN_SUBNET_Y).

The agent-attributes attribute returns a union of identification attributes for the reporting Agent 129. In the preferred embodiment within a credential specification, assertions about agent attributes may not be mixed with
20 assertions about any other credential attributes.

Table X herein below lists all the operations used in a preferred embodiment to make assertions about attributes.

Operation	Description
<i>absent</i>	Whether (true) the attribute denoted by the operand does not have a value in the protocol event
<i>and</i>	Logical AND of a list of boolean expressions, its operands
<i>eq</i>	Whether (true) two operands have the same value
<i>ge</i>	Whether (true) the first operand's value is greater than, or equal to, the second's
<i>gt</i>	Whether (true) the first operand's value is greater than the second's
<i>has</i>	Whether (true) the certificate chain defined by the first operand's value contains a certificate that satisfies the second operand (a credential name)
<i>hash</i>	Computes a digest of the second operand's value using the hashing function defined by the first operand; returns the hash as a hexadecimal string
<i>ip-mask</i>	<p>Whether (true) the first operand's value is included in the set of IP addresses defined by the second operand, an IP address prefix [RFC2280]; an address prefix is represented as an IPv4 address (dotted-decimal format with four integers) followed by the character slash / followed by an integer in the range from 0 to 32. The latter denotes the number of high-order bits from the preceding address that constitute a subnetwork address. If the subnetwork address bits match exactly the corresponding bits in the first operand's value, the operation returns true.</p> <p>The following are valid address prefixes: 128.9.128.5/32, 128.9.0.0/16, 0.0.0.0/0; the following address prefixes are invalid: 0/0, 128.9/16 since 0 or 128.9 are not dotted-decimal strings containing four integers.</p>
<i>ip-range</i>	Whether (true) the first operand's value is included in the set of IPv4 addresses defined by an IP address range whose lower bound is the operand with the lower value and whose upper bound is the operand with the higher value; the three operand values are taken as 32-bit unsigned integers and, if the first operand value falls within the inclusive numerical range defined by the two other operand values, the operation returns true
<i>le</i>	Whether (true) the first operand's value is less than, or equal to, the second's
<i>lt</i>	Whether (true) the first operand's value is less than the second's
<i>member</i>	Whether (true) the first operand's value is a member of the set defined by the second operand (a union)
<i>not</i>	Logical negation of its operand's value
<i>or</i>	Logical OR of a list of boolean expressions, its operands
<i>prefix</i>	Whether (true) the string that constitutes the first operand's value includes, starting at the first character, the string defined by the second operand
<i>present</i>	Whether (true) the attribute denoted by the operand has a value in the protocol event
<i>range</i>	Whether (true) the first operand's value is within the inclusive numerical range defined by the values of the second a third operands; the range comprises the set of values between the lower operand value and the higher

Operation	Description
<i>root</i>	Whether (true) the certificate chain defined by the first operand's value has, as its root, a certificate that satisfies the second operand (a credential name)
<i>substring</i>	Whether (true) the string that constitutes the first operand's value includes the string defined by the second operand

It is noted that the list of operations included in Table X is by no means complete. In other embodiments, the set of operations is expanded or reduced to reflect the set of protocols and features supported by the Policy

5 Monitoring System.

In the preferred embodiment credentials may be combined with other credentials or with additional assertions. Consider the following example:

```

10  ( credential Credentials_Example_2
      ( assertion
        ( or
          Credentials_Example_1
          ( and
15      ( ip-mask ip-address 207.5.0.0/16 )
        ( range ip-port 25 443 )
          )
        )
      )
20 )

```

The example herein above defines purported credentials that will be satisfied if either `Credentials_Example_1` is satisfied or if the presented credentials' IP address falls within the subnetwork defined by the address

25 prefix 207.5.0.0/16 and if the IP port is between 25 and 443, inclusive.

In the preferred embodiment the absence of an assertion about a specific attribute in a credential specification indicates that its value is to be ignored in considering the presented credentials. In the preferred embodiment, it is often useful to indicate that a particular attribute must or must not be specified in the presented credentials, irrespective of the attribute's value, if any. The operations `absent` and `present` accomplish this, as illustrated by the following examples:

```
10  ( credential Credentials_Example_3
      ( assertion
        ( and
          // http-username must exist, but don't care about its
value
          ( present http-username )
15    // the absence of an assertion about http-password
          indicates
          // that its presence or absence is irrelevant
        )
      )
20 )

( credential Credentials_Example_4
  ( assertion
    ( and
25    // an X.509 certificate must not have been presented
    ( absent der-cert )
    )
  )
30 )
```

Conditions

In the preferred embodiment a condition defines a constraint upon a protocol event 130. Said condition comprises a logical expression containing one or more assertions about attributes of the protocol event. Policy rules use

conditions to specify particular constraints that must or must not be satisfied by the protocol event 130.

Table Y lists attributes of a protocol event 130 that may be used when
5 formulating conditions. For each attribute the table shows protocols for which the attribute is defined, as well as the operations which can take the attribute as an operand.

10

Table Y

Attribute Name	Applicable Protocols	Description	Compare Operations
<i>auth-status</i>	SSL	The status of an authenticated session at the end of the authentication handshake	eq, member
<i>encipher-keysize</i>	SSL	The size of the key used for data encipherment (e.g., size of an IDEA key)	eq, member, gt, ge, lt, le, range
<i>http-cookie</i>	HTTP	Takes as an argument the name of a cookie in the request header and returns its value(s) as a union of strings	member
<i>http-req-hdr</i>	HTTP	Takes as an argument the name of a client request header and returns its value	eq, member, substring, prefix
<i>http-resp-hdr</i>	HTTP	Takes as an argument the name of a server response header and returns its value	eq, member, substring, prefix
<i>http-set-cookie</i>	HTTP	Takes as an argument the name of a cookie in the response header and returns its value(s) as a union of strings	member
<i>http-status-code</i>	HTTP	The status code returned on HTTP responses (aka response code)	eq, member, gt, ge, lt, le, range

Attribute Name	Applicable Protocols	Description	Compare Operations
<i>icmp-gateway-address</i>	ICMP	The IP address of the gateway host on a redirect message	eq, member, ip-mask, ip-range
<i>icmp-nested-address</i>	ICMP	The IP address carried in a destination unreachable message	eq, member, ip-mask, ip-range
<i>icmp-nested-port</i>	ICMP	The port number carried in a destination unreachable message	eq, member, gt, ge, lt, le, range
<i>initiator-access-rate</i>	all-protocols	The rate at which the current active principal has been the initiator of communications, over a predefined (configurable) period of time	eq, member, gt, ge, lt, le, range
<i>initiator-auth-keysize</i>	SSL	The size of the key used for initiator authentication and/or digital signatures (e.g., size of public key modulus)	eq, member, gt, ge, lt, le, range
<i>initiator-violation-rate</i>	all-protocols	The rate at which the current active principal has been the initiator of security policy violations, over a predefined (configurable) period of time	eq, member, gt, ge, lt, le, range
<i>ke-keysize</i>	SSL	The size of the key-encipherment key (e.g., size of public key modulus)	eq, member, gt, ge, lt, le, range
<i>protocol-version</i>	all-protocols	The version of the protocol	eq, member, gt, ge, lt, le, range
<i>ssl-ciphersuite</i>	SSL	The negotiated ciphersuite	eq, member
<i>target-access-rate</i>	all-protocols	The rate at which the current passive principal has been the target of communications, over a predefined (configurable) period of time	eq, member, gt, ge, lt, le, range
<i>target-auth-keysize</i>	SSL	The size of the key used for target authentication and/or digital signatures (e.g., size of public key modulus)	eq, member, gt, ge, lt, le, range
<i>target-violation-rate</i>	all-protocols	The rate at which the current passive principal has been the target of security policy violations, over a predefined (configurable) period of time	eq, member, gt, ge, lt, le, range

It is noted that the list of condition attributes included in Table Y is by no means complete. In other embodiments, the set of attributes is expanded or reduced to reflect the set of protocols and features supported by the Policy Monitoring System.

5

In the preferred embodiment operations listed in Table Y may be used to build assertions about condition attributes.

In the preferred embodiment condition attributes cannot mix with those from different protocol sets in a condition specification. A condition used in a policy rule for a protocol that is incompatible with the condition attributes in the condition object is considered an error and is flagged by the policy compiler. For example, it is illegal to use `ssl-ciphersuite` in a condition referenced by a policy rule for HTTP.

15

Following are some examples:

```
( group Strong_RSA_Ciphersuites  ciphersuite_t
  ( description "Strong ciphers with RSA key exchange" )
  ( union  SSL_RSA_WITH_RC4_128_MD5
           SSL_RSA_WITH_RC4_128_SHA
           SSL_RSA_WITH_IDEA_CBC_SHA
           SSL_RSA_WITH_3DES_EDE_CBC_SHA
           SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
           SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  )
)

( condition SslV3StrongCiphers           // condition <name>
  ( assertion
    ( and
```

```

      ( ge protocol-version \ version 3.0 / ,
      ( member ssl-ciphersuite Strong_RSA_Ciphersuites )
      ( ge ke-keysize 768 )
      ( ge target-auth-keysize 1024 )
5      )
    )
  )

10  ( condition HackerTripwire
      ( assertion
        ( ge initiator-violation-rate 10 )
      )
    )

15  ( condition ProtectSSL
      ( assertion
        ( and SslV3StrongCiphers HackerTripwire )
      )
    )

20

```

Herein above, the condition SslV3StrongCiphers can be used with an
 SSL protocol event to ensure that SSL 3.0 or higher is used, that the
 negotiated ciphersuite is one of the strong RSA-based ciphersuites, that the
 RSA key-encipherment key has a modulus of no less than 768 bits, and that
 25 the RSA authentication key has a modulus of no less than 1024 bits.

Herein above, the condition HackerTripwire can be used with any protocol
 event 130 to ensure that the active principal 144 is not a potential attacker.
 The third condition, ProtectSSL, simply combines the first two.

Dispositions

In the preferred embodiment a disposition defines an outcome of a policy rule.

Each policy rule may have many possible outcomes depending on, for
5 example, constraints imposed on the protocol event.

See Table Z herein for a list of disposition codes and an explanation of their meanings in the preferred embodiment.

10

Table Z

Disposition Code	Description
OK	The network event conforms to the security policy
CONTINUE	Additional information is needed before determining whether or not the network event conforms to the security policy
ACCESS_DENIED	Access to the target resource is denied by the security policy
AUTHENTICATION_VIOLATION	Authentication between the communication parties does not conform to the requirements set out by the security policy
SECURITY_ATTACK	A security attack has been detected
SECURITY_QOS	The security quality of service parameters associated with a protocol

Disposition Code	Description
	event do not meet the requirements set out by the security policy
POLICY_ERROR	An error has been detected in the security policy specification

It is noted that the list of disposition codes included in Table Z is by no means complete. In other embodiments, the set of disposition codes is expanded or reduced to reflect the set of features supported by the Policy Monitoring
5 System.

Table AA herein below lists possible severity codes in the preferred embodiment.

10

Table AA

Severity Code	Description
CRITICAL	Critical security violation, e.g., the network is undergoing an active security attack
HIGH	High-severity security violation, e.g., attempt to access sensitive data
MEDIUM	Medium-severity security violation, e.g., attempt to access a protected (but not highly sensitive) resource

Severity Code	Description
WARNING	Low-severity security violation, e.g., an incorrect password was entered
MONITOR	A security violation was not detected but an unusual or potentially suspect network event has occurred, e.g., TELNET access to a public web server
INFORMATION	A perfectly valid network event is being reported for informational purposes only

It is noted that the list of severity codes included in Table AA is by no means complete. In other embodiments, the set of severity codes is expanded or reduced to reflect the set of features supported by the Policy Monitoring
5 System.

Table AB herein below lists possible agent directives in the preferred embodiment.

10

Table AB

Agent Directive	Description
DECRYPT	The Agent is instructed to decrypt all traffic at the current protocol layer
DISRUPT	The Agent is instructed to terminate and/or disrupt all subsequent traffic

Agent Directive	Description
	associated with this network event
LOG_TRAFFIC	The Agent is instructed to log all traffic at the current protocol layer

It is noted that the list of agent directives included in Table AB is by no means complete. In other embodiments, the set of agent directives is expanded or reduced to reflect the set of features supported by the Policy Monitoring System.

Following are examples of preferred embodiments of dispositions:

```

10 // Network event ok but should be logged
   ( disposition Ok_Monitor           // disposition <name>
     ( code OK )                     // disposition code
     ( log-directive                  // logging directive
       MONITOR                       // severity code
       "Monitored activity" )        // logging string
15 )

```

The Ok_Monitor disposition is used to dispose of a valid network event 130 while flagging a logging subsystem that this event should be logged at a low severity level (MONITOR).

```

20 // Decrypt SSL session data and continue processing network event
   ( disposition Continue_Decrypt
     ( code CONTINUE )
     ( agent-directive DECRYPT )
   )

```

25 The Continue_Decrypt disposition is used to inform the policy engine 102 that additional information is needed from the Agent 129 before determining a final disposition 136 for the network event 130 while, at the same time,

instructing an appropriate Agent to decrypt all traffic at a current protocol layer.

```

5  // access to target resource is denied
   ( disposition Access_Denied
     ( code ACCESS_DENIED )
     ( log-directive
       HIGH
       "Access denied" )
10 )

```

The Access_Denied disposition is used as a final disposition 136 for a network event 130. It denotes a policy violation.

A list of built-in dispositions of the preferred embodiment is provided herein
15 above in Table S.

Rules

In the preferred embodiment a rule object defines a policy rule. A policy rule governs a specific interaction, or set of interactions, between two
20 communicating entities. The policy engine 102 evaluates policy rules against protocol events to determine if the latter conform to the active security policy.

Following is an example of a policy rule according to a preferred embodiment of the invention:

```

25 ( rule Tcp_Ext2Int           // rule <name>
   ( description "Communications from external hosts" )
   ( agent Foo_Subnet_Monitor ) // the reporting agent
   ( protocol TCP )           // the protocol
30 ( action CONNECT )         // the protocol action
   ( initiator External_Hosts ) // the active principal

```

```

( target internal_hosts )           // the passive principal
( outcome
  ( immediate                       // the immediate outcome
    // if/ifnot <condition> <disposition>
5    ( if Catch_Suspect Security_Attack_Possible )
    ( if Catch_Attacker Security_Attack_Progress )
    ( default continue )           // using built-in disposition
  )
  ( final                           // the final outcome
10  ( default Ok_Monitor )
  )
)
)

```

15 In the preferred embodiment a policy rule comprises:

- 20 • *Agent* – represents Agent 129 that reported the protocol event 130. The Agent 129 is denoted by a credential name. The policy rule is only considered if this credential is satisfied by the credentials presented by the reporting Agent 129. In the example above, Foo_Subnet_Monitor is the name of a credential object identifying one or more Agents. This field is optional. If omitted, the rule applies to all Agents.
- 25 • *Protocol* – a protocol to which the rule is applicable. A protocol event 130 addresses one and only one protocol. This field is mandatory. Note that the special token `ignore` is used to denote a rule that applies to all protocols.
- 30 • *Action* – a protocol action to which this rule is applicable. Each protocol comprises one or more several distinct actions (e.g. connect, transfer-

data, release), some of which might be of interest to the security policy. A protocol event denotes one and only one protocol action. This field is mandatory. Note that the special token `ignore` is used to denote a rule that applies to all actions within the specified protocol.

5

- *Initiator* – represents the active principal 144 in the protocol event 130. The initiator 144 is denoted by a credential name or by the special tokens `absent` (credentials must not be presented in the protocol event), `present` (credentials must be presented but their actual value is unimportant) and `ignore` (credentials may or may not be presented). In the example herein above, `External_Hosts` is the name of a credential object identifying one or more TCP/IP hosts. This field is mandatory.

10

- *Target* – represents the passive principal 145 in the protocol event 130. The target 145 is denoted by a credential name or by the special tokens `absent`, `present` and `ignore`. In the example above, `Internal_Hosts` is the name of a credential object identifying one or more TCP/IP hosts. This field is mandatory.

15

- *Prerequisite* – (not shown in the example above) one or more rules that must be satisfied by a previous protocol event. Prerequisite rules are identified by names. Prerequisites are used to place additional constraints on an entire network event 130. See an example herein that illustrates the use of prerequisites in rules. It should be noted that if two or more rules are listed as prerequisites, the prerequisite is satisfied if *any* of the listed

20

25

rules taken in the order in which they are listed satisfies a previous protocol event. This field is optional.

- *Outcome* – the outcome section defines what to do with the protocol (or network) 130 event if the current policy rule is applied to the protocol event. That is, if the rule is selected by the policy engine 102 as the most suitable for the protocol (or network) event. Every policy rule must have a disposition that applies to the protocol event and another disposition that applies to the entire network event. In some cases these are one and the same. The policy engine 102 evaluates the outcome and produces a disposition for either the protocol or the network event. There are two outcomes defined:

- *Immediate* – an immediate outcome applies to the protocol event immediately. A policy rule may or may not include an immediate outcome. If it does, the outcome is evaluated as soon as the rule is selected for the protocol event. If it does not, there is an implied disposition for the protocol event, a built-in disposition *continue* (see Table S for the definition) which instructs the policy engine 102 to continue processing the network event. If the immediate outcome generates a disposition with a disposition code other than *CONTINUE*, this disposition becomes the disposition for the entire network event. In this instance, the final outcome, defined herein below, will not be evaluated.

- *Final* – an outcome that applies to the entire network event if this rule becomes a final rule evaluated for that event. The final outcome must be specified if the immediate outcome does not generate a final disposition for the network event. If it is not, an implied disposition for the network event, the built-in disposition *policy-error*, see Table S for the definition, denotes a policy specification error. The final outcome is evaluated when the policy engine determines that no additional protocol events are to be considered for the current network event. The final outcome must always generate a final disposition, *i.e.* a disposition with a disposition code of CONTINUE is not allowed in a final outcome.

In the preferred embodiment each outcome section comprises one or more conditional statements, each followed by a disposition. The purpose of conditional statements is to specify constraints upon a protocol event, or special conditions that, if satisfied, cause the generation of an alternate disposition for the protocol (or network) event. Conditional statements are evaluated in the order in which they are specified within the outcome section.

In the preferred embodiment a conditional statement starts with one of the following keywords:

- *if* – takes as arguments a condition and a disposition, each referenced by name. If the condition evaluates to TRUE, the disposition becomes the disposition for the protocol event.

- ifnot – takes as arguments a condition and a disposition, each referenced by name. If the condition evaluates to FALSE, the disposition becomes the disposition for the protocol event.
- default – takes a single argument, a disposition referenced by name.

5 It is equivalent to a condition that is always satisfied, thereby triggering the disposition that is its argument. This conditional statement is mandatory and must be the last conditional statement in an outcome.

10 The following examples illustrate the use of prerequisites in rules in a preferred embodiment. The first rule is the prerequisite.

```

( credential Host_A
  ( assertion
    ( and
      15      ( eq ip-address 207.5.63.8 )
      ( eq ip-port 80 )
    )
  )
)
20
( rule Access_Host_A
  ( protocol TCP )
  ( action CONNECT )
  ( initiator ignore )
  25  ( target Host_A )
  ( outcome
    ( final
      ( default Access_Denied ) // Access_Denied defined above
    )
  )
  30 )
)

```

Herein above, the rule `Access_Host_A` states that access to host A on port 80 by any host is denied, unless explicitly allowed by a rule at a higher protocol layer. Note the use of a final outcome, which is only evaluated if `Access_Host_A` becomes the applicable rule for the entire network event.

5 The implied disposition for the protocol event is CONTINUE.

This rule can be overridden by another rule at the HTTP layer stating that access is allowed to host A on port 80, as shown below:

```

10  ( rule Http_To_Host_A
      ( protocol HTTP )
      ( action ignore )
      ( initiator ignore )
      ( target ignore )
15  ( prerequisite Access_Host_A )    // reference to rule above
      ( outcome
        ( immediate
          ( default ok )              // using built-in disposition
        )
20  )
    )

```

The end result of the two policy rules herein above is to prevent all access to host A on port 80 unless that access is using HTTP over TCP/IP.

25

In the preferred embodiment a prerequisite rule is any rule that is selected for a previous protocol event. This includes rules in the same protocol layer. As an example, to ensure that a web server requires HTTP authentication before allowing access to a specific web page, use the following rules:

30

```

( credential Some_Url

```

```

    ( assertion
      ( prefix url "//myserver.com/Documents" )
    )
  )
5
  ( rule Host_A_Anon_Access
    ( protocol HTTP )
    ( action ( union GET POST ) )
    ( initiator absent )
10    ( target Some_Url )
    ( prerequisite Access_Host_A )    // from example above
    ( outcome
      ( final
        ( default Access_Denied )    // Access_Denied defined above
15      )
    )
  )

  ( condition Require_Auth
20    ( description "Check if server returned the Unauthorized response
      " "code" )
    ( assertion
      ( eq http-status-code 401 )
    )
25  )

  ( rule Check_Access_Denial
    ( protocol HTTP )
    ( action RESPONSE )
30    ( initiator ignore )
    ( target ignore )
    ( prerequisite Host_A_Anon_Access )
    ( outcome
      ( immediate
35        ( ifnot Require_Auth Access_Denied )
        ( default ok ) // using built-in disposition
      )
    )
  )
40

```

The example herein above shows that access to the document sub-tree identified by `Some_Url` requires the user be authenticated using basic HTTP authentication. The authentication is accomplished by means of the condition `Require_Auth` which, in the context of rule `Check_Access_Denial`, checks that the server returns an *Unauthorized* status code. If the server fails to do so, the `Access_Denied` disposition is generated. Note that the prerequisite constraint ensures that the rule `Check_Access_Denial` is only considered if the rule `Host_A_Anon_Access` is selected when the HTTP request event is evaluated, that is, requests where basic HTTP authentication is not used.

The Policy Specification Process

In the preferred embodiment the policy specification process comprises the following steps:

1) Identify communicating entities recognized by the security policy. The entities comprise physical networks and sub-networks, host machines, communication protocols, users, applications, services, and any other resources of interest.

2) Identify relationships between the communicating entities and define rules to control said relationships (e.g. host A may communicate with host B but not with host C).

3) Formally define communicating entities and entity relationships using the policy specification language (Fig. 1; 108) according to the invention. In a preferred embodiment a visual tool is used. In another embodiment a text-based editor is used. In the preferred embodiment the output of this step is a

policy specification in an advanced encoding format according to the invention.

4) Compile the policy specification with a Policy Compiler (Fig. 33, 101). In one embodiment, said compilation step is incorporated into a graphical policy editor, such that it is incorporated into said policy specification step. In another embodiment it is a distinct step. This step comprises:

- a) Checking the specification for errors of syntax or semantics;
- b) Checking the specification of credentials for errors (*e.g.* credentials that can never be satisfied);
- c) Checking the specification of conditions for errors (*e.g.* conditions that can never be satisfied);
- d) Checking the specification of rules for completeness and coverage;
- e) Ordering credentials based on their specificity (described in detail herein below);
- f) Ordering rules based on the credentials of their principals (described in detail herein below); and
- g) Resulting in an annotated policy specification (Fig. 1, 109) represented by a text file (Fig. 33 105).

The annotated policy specification 105 is suitable for loading into the policy engine 102 for evaluation of one or many network events 130, or back into the graphical policy editor for visualization and further refinement.

Evaluation of Rules

This section describes how policy rules are organized and evaluated according to the invention.

Policy Evaluation Model

The policy specification language 108 alone does not describe how the policy engine 102 evaluates policy rules. In the preferred embodiment of the invention, a security administrator that writes the policy specification 105 and
5 the policy engine 102 that enforces the policy specification 105 share a common view of the evaluation procedure. The evaluation of policy rules is deterministic.

In the preferred embodiment of the invention the basic policy specification
10 language 108 is augmented to convey information about how rules are ordered for purposes of evaluation, *i.e.* which rules are evaluated first and which rules are selected for any given network event. The augmented language is a superset of the basic specification language 108 and it is hereinafter referred to as the annotated specification language 109.

15

In one embodiment the security administrator uses the annotated specification language 109 using a visual tool, such as a graphical policy editor to determine how the policy rules are interrelated, their hierarchical relationships and how they will be evaluated. This step is crucial to determining whether
20 the specified policy correctly reflects the desired security policy and to identifying areas where the policy specification needs refinement.

In the preferred embodiment the policy engine 102 uses the annotated language 109 to organize the policy, after having converted it to an internal
25 representation in a manner best suited for the efficient evaluation of network events.

In the preferred embodiment the policy engine 102 receives protocol events in proper sequence. Protocol events for protocols lower in the protocol stack are received before protocol events for protocols higher in the stack. This
5 sequencing is important because the policy engine 102 must make a policy decision about, for example, a TCP connection, before it makes a decision about an SSL session that uses that TCP connection.

Data about a specific protocol event may not arrive all at once. For example,
10 when evaluating an SSL session the policy engine 102 first receives the server certificate and negotiated ciphersuite before receiving a client certificate or a message indicating that none was provided. In a preferred embodiment, the policy engine 102 uses incomplete information about a protocol event in order to collect a set of possible policy rules applicable to
15 that event. However, for the sake of simplicity, the remainder of this document assumes that Agents convey information about protocol events in an atomic manner.

In the preferred embodiment for every protocol event the policy engine 102
20 selects a policy rule applicable to that event. Every policy rule is associated with a specific protocol and action or a set of protocols and actions. Therefore only the set of rules relevant to the protocol event is considered. Of that set, several rules can be satisfied by the event. In the preferred embodiment a policy rule is satisfied by a protocol event if the following holds true:

25

- 1) The credentials of the Agent 129 reporting the event match the rule's agent credentials (if any), defined as a set of attribute-value assertions.
- 2) The rule's protocol specifier matches the protocol identifier in the protocol event.
- 3) The rule's action specifier matches the action identifier in the protocol event.
- 4) The rule's prerequisite clause is satisfied (details described herein below).
- 5) The credentials of the initiator 144 and target 145 principals in the protocol event satisfy the rule's corresponding credentials, defined as a set of attribute-value assertions.

In the preferred embodiment when several rules are satisfied by a protocol event, the policy engine 102 selects a rule that is most specific to the protocol event. The specificity of a policy rule is determined by the specificity of the credentials associated with the policy rule, as well as the specificity of the rule's protocol, action and prerequisite specifiers. For example, a rule that targets one single protocol is more specific than a rule that targets all protocols. In another example, a rule that specifies a prerequisite is more specific than a rule that does not.

In the preferred embodiment the specificity of a credential specification is determined by the set relationships of said specification with other credential specifications. Following are examples of credential specifications:

- A: all principals with blue eyes
- B: all principals with blue eyes and black hair

- C: all principals with black hair

B defines the intersection of A and C, *i.e.* B is a subset of both A and C. Thus, B is more specific than either A or C.

5

According to the invention, in general, the more data described about a principal the more specific are the credentials. In the preferred embodiment, some attributes of a principal's credentials have more importance than do other attributes of the credentials. In the preferred embodiment the importance of an attribute is represented by its weight. The attribute weight is
10 determined by its role as a discriminator of principals. For example, an attribute that yields a small set of principals has more weight than an attribute that yields a larger set of principals. In the hair and eye color example herein above, it is arbitrary to give a higher weight to eye color versus hair color or to
15 give both hair and eye color the same weight. Assigning an attribute weight is easier because typically protocol credentials are structured hierarchically. For example, in the TCP protocol, the IP address attribute has clearly more weight than the IP port attribute because the number of principals with a given IP address is generally much smaller than the set of principals with a given port
20 number.

In the preferred embodiment attributes that comprise a set of credentials are ranked by weight and the combined weight of all attributes in a credential specification is considered in determining a relative specificity of said
25 specification.

In the preferred embodiment a policy specification has sets of credentials each of which are ranked at a same specificity level, thereby rendering many policy rules that are applicable to a given protocol event. Herein below is provided a section describing a number of practical guidelines for good policy development that minimize herein above ambiguities.

Fig. 37a is a schematic diagram of the preferred embodiment in which a network event 130 comprises M protocol events at different protocol layers, and in which the network event 130 has an associated network event disposition 136.

Fig. 37b is an algorithm showing how the M protocol events at different protocol layers of the network event 130 result in pending rules with or without immediate outcomes and, finally, a final disposition for the network event 136. For clarity, the algorithm assumes that the policy engine 102 always finds a policy rule applicable to a given protocol event, that at least a first Protocol Event (1) exists, and that the algorithm ends when the Agent 129 informs the policy engine 102 that no further protocol events will be generated. These assumptions are for clarifying purposes only and do not limit the invention in any way.

The algorithm begins with $j=1$ (3700) and with the policy engine 102 receiving Protocol Event (1) from the Agent 129 (3701) (3702).

Once a most specific policy rule is selected for a given protocol event (3703), the policy engine 102 consults an outcome clause (3704) determining if an

immediate outcome is applied to the protocol event. In the preferred embodiment an immediate outcome applies to a protocol event while a final outcome applies to a network event (130).

- 5 In the preferred embodiment an immediate outcome is executed when it is specified. The immediate outcome can evaluate constraints (*i.e.* conditions) against a protocol event, produce a set of agent directives (*e.g.* instructing the Agent 129 to decrypt all subsequent traffic), and produce a final disposition (3706) for the protocol event rendering said disposition for the entire network
- 10 event. When a disposition of an immediate outcome is not a final disposition, a special disposition code, CONTINUE, is used as an indicator. All disposition codes other than CONTINUE denote final dispositions.

- In the preferred embodiment when an immediate outcome does not produce a
- 15 final disposition the associated selected policy rule becomes a pending policy rule for the related network event (3707). The policy engine 102 then waits for further protocol events of the network event 130 from the Agent 129 (3708) and (3701). In this embodiment, said pending policy rule is overridden by subsequent policy rule selected for a protocol event higher in the associated
- 20 protocol stack (3707).

- In the preferred embodiment policy evaluation ends in one of two cases. First case is when no further rules in the policy apply to a network event (*e.g.* a highest protocol in the stack is reached). Second case is when the Agent 129
- 25 informs the policy engine 102 that no further protocol events will be generated (3702) (3709) (3706) (3710). In either case, a policy decision is then

expected for the entire network event. The policy engine 102 selects a pending policy rule for a protocol highest in the protocol stack and executes the final outcome defined for that rule (3711). In the preferred embodiment constraints are evaluated against the entire network event. In the preferred
5 embodiment a final outcome always produces a final disposition (3709) which becomes a disposition for the network event (3706).

In the preferred embodiment a protocol event must result in a selection of a policy rule (pending or final). When a policy rule applicable to a given protocol
10 event is not found, the policy engine 102 produces a special disposition identifying a policy specification error. See the default policy rule in Table S.

Ordering of Credentials

In the preferred embodiment credentials are ordered based on a combined
15 weight of all attribute-value assertions that make up a credential specification.

In the preferred embodiment computing a weight of an attribute-value assertion of an attribute requires the following two steps:

- 20 1) Assigning a ranking value to the attribute. Attributes that are listed in a credential specification are ranked against each other. Ranking is based on a value of the attribute as a discriminator of principals identified by the credentials. If the presence of the attribute in a credential specification generally yields a smaller set of principals than the presence of another
25 attribute, then the former has a higher ranking than the latter.

2) Assigning a ranking value to an assertion type of the attribute. An assertion type is used to make an assertion about the value of an attribute (e.g. eq, substring, range). Following are five assertion types, in decreasing ranking order:

5

a) *Absent* – an assertion not satisfied by any attribute value. The attribute is absent from the presented credentials. In one embodiment said assertion type typically is used to require the absence of an entire set of credentials (e.g. “no SSL client certificate”).

10

d) *Single-value* – an assertion satisfied by a single attribute value (e.g. “hair color is blue”).

e) *Multi-value* – an assertion satisfied by any value within a set of attribute values (e.g. “port number in the range of 3200 to 4200”).

15

d) *Present* – an assertion satisfied by any attribute value, wherein the associated attribute must be present in associated presented credentials.

e) *Ignore* – an assertion always satisfied, irrespective of whether the associated attribute is present or absent from associated presented credentials. This is a “don’t care” matching rule.

20 Table AC herein below shows the preferred embodiment assertion types for all operations that operate on attributes to build assertions. In the preferred embodiment when a credential specification does not include any assertions about a particular attribute then the assertion type for that attribute is ignore.

25

Table AC

Operation	Assertion Type
-----------	----------------

Operation	Assertion Type
<i>absent</i>	Absent
<i>eq</i>	Single-value
<i>ge</i>	Multi-value
<i>gt</i>	Multi-value
<i>has</i>	Multi-value
<i>ip-mask</i>	Multi-value
<i>ip-range</i>	Multi-value
<i>le</i>	Multi-value
<i>lt</i>	Multi-value
<i>member</i>	If the union has a single terminal member, the assertion type is single-value, otherwise it is multi-value
<i>prefix</i>	Multi-value
<i>present</i>	Present
<i>range</i>	Multi-value
<i>root</i>	Multi-value
<i>substring</i>	Multi-value

In the preferred embodiment assertions in a credential specification often are combined using logical operators and, or and not. For example,

```

5      ( credential Credentials_Example_1
      ( assertion
        ( and
          ( eq ip-address 207.5.63.8 )
10      ( or
          ( eq ip-port 80 )
          ( gt ip-port 443 )
        )
      )
15  )
  )

```

In the preferred embodiment a weight assigned to a credential specification is derived from a combined weight of all assertions the credential specification comprises. An algorithm herein below is used recursively to compute a combined weight of a set of assertions operated on by a logical operator:

5

A. An operator `not` does not affect the weight of its operand.

B. An operator `and` creates a union of weights of all its operands. The weights are sorted in decreasing order of attribute rank. If multiple assertions
10 are made about a particular attribute, use a weight of a most specific assertion and discard all other weights for that attribute. If multiple **distinct assertions** (*i.e.* not identical or equivalent) are made about a particular attribute at a **same level of specificity**, the assertions are enumerated. In general, the higher a number of distinct assertions made about an attribute
15 the more specific is a credential specification. For example, the two assertions "hair is not black" and "hair is not brown" when combined in a union are more specific than either individual assertion.

C. An operator `or` results in a selection of an operand with a lowest weight.
20 In addition said combined weight is penalized, such that it weighs less than the associated assertion with the lowest weight. If two or more assertions of equal weight are combined with `or`, the combined weight is lower than that of either or any individual assertion. The rationale behind the penalty is that, in general, combined assertions yield a larger set of principals (*i.e.* is less
25 specific) than each assertion by itself. The weight penalty is associated with the entire credential specification, not with an individual assertion or set of

assertions. Thus, for every instance of the operator `or` in the credential specification, the weight penalty is incremented by one.

In the preferred embodiment a 3-tuple represents a weight of all attribute-value assertions about a specific attribute within a credential specification.
5 Elements in the 3-tuple are:

- Attribute rank
- Assertion type rank
- 10 • Attribute assertion count

In the preferred embodiment the 3-tuple is represented by a weight S-expression in the annotated specification language. A syntax of this expression is:

15

```
( weight <attribute> <assertion-type> <assertion-count> )
```

In the preferred embodiment ranking of assertion types is fixed and defined by the Table AD following:

20

Table AD

Assertion Type	Ran k
absent	4
single-value	3

Assertion Type	Ran k
multi-value	2
present	1
ignore	0

In the preferred embodiment ranking of an attribute is configurable by a security administrator and must be defined prior to a compilation of a policy specification. Attribute ranking is communicated to the policy compiler in a variety of ways. Table AE herein below shows a preferred embodiment of proposed rankings for attributes used in credentials for all supported protocols. Said rankings are assumed in examples used throughout the remainder of this document. It is noted that a credential attribute agent-attributes cannot be used in a specification of an initiator or target credential and therefore need not be ranked. It is further noted that the special assertions *true* and *false*, which are allowed by the policy specification language's grammar in the preferred embodiment, do not apply to any specific attribute and, thus, are assigned a special weight consisting of a zero valued attribute rank, a zero valued assertion type rank and a zero valued attribute assertion count.

Table AE

Protocol/Action	Attribute	Rank
IP/ASSOCIATION UDP/ASSOCIATION ICMP/ASSOCIATION TCP/CONNECT	mac-address	3
	ip-address	2
	ip-port	1
SSL/HANDSHAKE	der-cert	5
	x509-subject	4
	x509-issuer	3
	x509-cert-path	2
SSL/HANDSHAKE	cert-status	1
HTTP/GET HTTP/POST HTTP/HEAD	http-username	3
	http-password	2
	url	1

In the preferred embodiment an attribute assertion count starts at zero for a first assertion and is incremented monotonically for all subsequent assertions.

That is, the count enumerates additional assertions for the attribute. In the

5 preferred embodiment the assertion count is omitted from the weight S-expression when said count is zero.

In the preferred embodiment a weight S-expression is omitted when an assertion type is ignore.

10

In the preferred embodiment the three elements of a 3-tuple are used in sorting a collection of 3-tuples. The attribute rank as a primary key, the assertion type rank as a secondary key, and the attribute assertion count as a tertiary key produce an ordered list of 3-tuples sorted in decreasing order of

15 rank and count. In the preferred embodiment said sorted list is used to rank credential specifications against each other. The sorting algorithm is described using pseudo-code in Table AF herein below:

Table AF

20

```
Sort_3tuples: subroutine ( 3tuple A, 3tuple B )
  begin
    if A.attribute_rank > B.attribute_rank
      return (A is greater than B);
    else if A.attribute_rank < B.attribute_rank
      return (A is less than B);
    else // same attribute rank
      if A.assertion_type > B.assertion_type
        return (A is greater than B);
      else if A.assertion_type < B.assertion_type
        return (A is less than B);
```

25

30

```
else // same assertion type
  if A.assertion_count > B.assertion_count
    return (A is greater than B);
  else if A.assertion_count < B.assertion_count
5    return (A is less than B);
  else // same assertion count
    return (A is equal to B);
end
```

10 A weight penalty is represented by the following S-expression in the annotated specification language:

```
( weight-penalty <penalty-count> )
```

15 where <penalty-count> is an integer representing a number of or operators in a credential specification.

Thus, Credentials_Example_1 herein above is annotated as follows:

```
20 ( weight ip-address single-value )
   ( weight ip-port multi-value )
   ( weight-penalty 1 )
```

In the preferred embodiment a credential specification can combine previous
25 credential specifications with each other or with additional assertions. In the preferred embodiment rules for a combination of assertions with logical operators apply equally to a combination of credential specifications. For example:

```
30 ( credential Credentials_Example_2
```

```

    ( assertion
      ( eq ip-address 207.5.63.22 )
    )
  )
5
  ( credential Credentials_Example_3
    ( assertion
      ( and
        Credentials_Example_2
10      ( gt ip-port 42 )
      )
    )
  )
15
  ( credential Credentials_Example_4
    ( assertion
      ( and
        ( or
          Credentials_Example_1
20      Credentials_Example_3
        )
        ( lt ip-port 1025 )
      )
    )
25  )

```

The weight of Credentials_Example_2 is:

```
( weight ip-address single-value )
```

30 The weight of Credentials_Example_3 is:

```
( weight ip-address single-value )
( weight ip-port multi-value )
```

In the embodiment to compute the weight of Credentials_Example_4 first
 35 compute a weight of the or expression. Credentials_Example_1 is
 selected as having a lowest weight because of an associated weight penalty.

Furthermore, the or expression in Credentials_Example_4 increases the weight penalty further, yielding:

```

5      ( weight ip-address single-value )
      ( weight ip-port multi-value )
      ( weight-penalty 2 )

```

In the embodiment the and expression adds an additional, distinct, assertion about ip-port. The assertion is of the same type as one currently selected because they are both multi-value assertions. The assertion count for ip-

```

10 port is incremented, yielding:
      ( weight ip-address single-value )
      ( weight ip-port multi-value 1 )
      ( weight-penalty 2 )

```

15 In the embodiment a ranking algorithm for comparing and ordering credentials is implied in the example previously described herein above. Following in Table AG is an associated algorithm using pseudo-code:

Table AG

```

20 Rank_credentials: subroutine ( credentials A, credentials B )
    begin
        Let X be a sorted list of 3-tuples that describe the weight of
        credential specification A;
25        Let Y be a sorted list of 3-tuples that describe the weight of
        credential specification B;

        // compare 3-tuples
30        for all i in X do
            R = Sort_3tuples(X[i],Y[i]); // defined above
            if R is greater-than
                return (A ranks higher than B);

```

```

    else if R is less-than
        return (A ranks lower than B);
    else
        continue; // 3-tuples are equal
5      end

    // X and Y are the same; compare weight penalties

    if A.weight_penalty < B.weight_penalty
10      return (A ranks higher than B);
    else if A.weight_penalty > B.weight_penalty
        return (A ranks lower than B);
    else
        return (A and B have the same rank);
15  end

```

The following Table AH ranks example credentials according to the preferred embodiment using the algorithm herein above. A weight column shows 3-tuples using a format *W:x,y,z*, wherein *x* is an integer value for an attribute rank (Table AE), *y* is an integer value for an assertion type (Table AH), and *z* is an assertion count. A weight penalty is shown as *P:x*, wherein *x* is a penalty count. It is noted that the higher a rank of a credential specification, the more specific it is. For completeness, the table includes ranking for built-in credentials denoted by absent, present and ignore. Said built-in credentials

25 make assertions about and in the order of an absence, presence, and irrelevance of any credentials presented by a protocol event. It is noted that in the preferred embodiment ignore and present always rank lower and absent higher than do any user-defined credentials.

30 Table AH

Name	Weight	Rank
absent (built-in)	W:*,5	6
Credentials_Example_4	W:2,3 W:1,2, 1 P:2	5
Credentials_Example_3	W:2,3 W:1,2	4
Credentials_Example_1	W:2,3 W:1,2 P:1	3
Credentials_Example_2	W:2,3	2
present (built-in)	W:*,1	1
ignore (built-in)	W:*,0	0

Ordering of Rules

In the preferred embodiment policy rules must be organized such that when two or more rules are satisfied by a protocol event, the most specific rule for that event is selected. The specificity of a policy rule is fully determined by the specificity of the credentials it uses.

In the preferred embodiment policy rules are organized as follows:

1) Rules are segregated by protocol. For example, rules that apply to a TCP protocol are separated from those that apply to a SSL protocol.

2) Within each protocol group, rules are segregated by action. For example,
5 rules that only apply to a TCP CONNECT action are separated from those that only apply to a TCP CLOSE action.

3) Within each protocol-action group, rules are ranked by the specificity of their respective credentials. The ranking algorithm is:

10

a) Create a 2-tuple from a ranking order of an initiator credential and a target credential. The first element in said 2-tuple is a highest-ranking value and the second element a lowest. That is, said 2-tuple is defined as
(MAX(I,T), MIN(I,T)), wherein I and T are the ranking values for the
15 initiator and target credentials, respectively.

15

b) Sort the rules in increasing ranking order using the first element in the 2-tuple as the primary sorting key and the second element as the secondary key. Rules with identical 2-tuples are given the same ranking
20 number. The rule or rules with the highest-ranking number is the most specific rule for the protocol group.

20

In the preferred embodiment and because rules are ranked directly from the ranking of their credentials, a special representation is not provided in the
25 annotated specification language for the ranking of the policy rules.

Following is an example using credentials from herein above:

```
( rule Rule_Example_1
  ( protocol TCP )
5   ( action CONNECT )
  ( initiator Credentials_Example_2 ) // ranked #2
  ( target Credentials_Example_1 )    // ranked #3
  ( outcome
    ...
10  )
)

( rule Rule_Example_2
  ( protocol TCP )
15  ( action CONNECT )
  ( initiator Credentials_Example_1 ) // ranked #3
  ( target Credentials_Example_2 )    // ranked #2
  ( outcome
    ...
20  )
)

( rule Rule_Example_3
  ( protocol TCP )
25  ( action CONNECT )
  ( initiator Credentials_Example_2 ) // ranked #2
  ( target Credentials_Example_4 )    // ranked #5
  ( outcome
    ...
30  )
)
```

Table AI herein below shows how said rules are ranked according to the invention.

Table AI

Name	Credentials	Rule
	Rank	Rank
Rule_Example_3	(T:5, I:2)	2
Rule_Example_1	(T:3, I:2)	1
Rule_Example_2	(I:3, T:2)	1

It is noted that Rule_Example_1 and Rule_Example_2 are ranked at the same specificity level. This does not represent a problem because the
5 respective initiator and target credential sets are non-intersecting and used in different roles.

In the preferred embodiment it is possible for two or more rules at a same specificity level to be satisfied by a single protocol event. During policy
10 specification a security administrator disambiguates the evaluation of rules with the same specificity level by forcing a ranking order among them. Forcing a ranking order is done by specifying that one rule is ranked above another rule and is termed forced ranking. Forced ranking is expressed by means of the following S-expression:

15 (rank-above <rule-name>)

For example, to give Rule_Example_2 precedence over Rule_Example_1, the following S-expression is added to a definition of Rule_Example_2:

(rank-above Rule_Example_1)

In the preferred embodiment after performing the standard ranking algorithm herein above, the policy engine 102 evaluates all rank-above expressions
5 and reassigns ranking numbers to each rule accordingly. In the preferred embodiment it is important to note that forced ranking does not force a ranking of an affected rule to a level of a more specific rule higher in the ranking order. Instead a new ranking level is created for the affected rule and all other ranking numbers of more specific rules are incremented accordingly.

10

For example, Rule_Example_2 herein above is given ranking number 2 and the ranking number of Rule_Example_3 herein above is incremented from 2 to 3.

15

In the preferred embodiment forced ranking is applied to any rule and is not limited by rules having only non-unique ranking numbers. In this embodiment security administrators are cautioned not to use said forced ranking feature unless absolutely necessary. Its misuse may result in a policy specification that is both difficult to manage and difficult to evaluate. In the preferred
20 embodiment runtime conflicts in the evaluation of rules (i.e. when a protocol event is satisfied by multiple rules) typically can be solved by redesigning credentials upon which said rules are based. Useful tips are provided herein below.

Evaluation Algorithm

In the preferred embodiment the policy engine 102 applies a policy evaluation algorithm to each incoming protocol event. The algorithm results in a selection of a policy rule applicable to the protocol event and may produce an immediate or final disposition.

Following is a step-by-step description of the evaluation algorithm according to the preferred embodiment. It is noted that the evaluation procedure described herein below is in conceptual form and does not take into account any possible runtime optimizations:

- 7) Select a set of rules applicable to an Agent reporting an event;
- 8) From said set, select a second set of rules applicable to an associated examined protocol.
- 9) From said second set, select a third set of rules applicable to an associated examined protocol action.
- 10) Starting with a most specific policy rule in said third set and descending to a least specific rule find a policy rule satisfied by said protocol event. A matching algorithm according to the preferred embodiment is as follows:
 - a) If one or more orderly listed prerequisite rules are specified, ensure at least one of said prerequisite rules is satisfied by a previously

processed protocol event. In the preferred embodiment a prerequisite rule is satisfied if it is a pending policy rule for the protocol event.

5 b) Match initiator and target credentials in the policy rule against the corresponding initiator and target credentials presented in the protocol event.

11) If a policy rule satisfying the protocol event is not found the policy engine 102 generates a disposition for the network event indicating that a policy specification error was encountered. Effectively the processing of the network event thereby terminates.

12) If a policy rule satisfying the protocol event is found, the policy engine 102 checks for other rules having a same ranking number and also satisfying the event. If such rules are found the policy engine 102 uses the following algorithm in the preferred embodiment to select a single applicable rule:

20 a) Rules that specify all protocols (*i.e.* using `ignore` or `present`) are less specific than rules that explicitly list a set of one or more protocols.

 b) Rules that specify all actions (*i.e.* using `ignore` or `present`) are less specific than rules that explicitly list a set of one or more actions.

25 c) Rules that have prerequisites are more specific than rules that do not have prerequisites. Rules that specify a higher-ranking prerequisite are more specific than rules that specify a lower-ranking prerequisite.

In the preferred embodiment a ranking relationship is relevant only if both prerequisite rules belong to a same protocol-action group.

d) If thereafter a single rule is determined as more specific than the others it is selected for the protocol event. If more than one rule remains the policy engine 102 sorts the remaining rules in increasing lexical order by name and selects a first rule from the sorted rules having an immediate disposition indicating in decreasing order of precedence:

- i) a policy violation (any disposition code other than OK or CONTINUE);
- ii) CONTINUE (allows other rules to examine further the network event); and
- iii) OK

The outcome of the policy evaluation algorithm herein above is a policy rule that satisfies the protocol event. If an immediate outcome is specified for that rule, it is executed, producing a disposition for the protocol event. If the disposition comprises a final disposition code (any code other than CONTINUE), the disposition is also the final disposition for the network event.

Otherwise in the preferred embodiment the selected policy rule is a pending policy rule for the network event. In absence of any further protocol events the pending policy rule is promoted to selected policy rule. A final outcome of the selected policy rule is executed producing a final disposition for the network event.

Policy Specification Guidelines

Provided herein below in Table AJ are a number of practical guidelines coupled to the preferred embodiment for the development and specification phases of a security policy. Adhering to the guidelines ensures efficient and accurate evaluation of a policy by the policy engine 102. It is intended to incorporate the guidelines into a graphical policy editing invention using wizards, policy templates and other UI mechanisms that among other uses simplify and direct the policy specification process.

10

Table AJ

Rule #1: Work on group relationships

The first step in policy specification is identifying the communicating entities and resources that interact with each other over the network, that is to say, specifying the credentials for both initiator and target principals. Defining groups in relation to each other can significantly enhance the ranking of credentials. This is best done by:

- 20 ◦ Defining all large groups first (e.g. all hosts in the corporate network, all valid certificates).
- Defining all other groups by subsetting larger groups (e.g. all hosts in the marketing subnetwork, all certificates issued by the corporate CA, all
25 revoked certificates).

The process of defining a group as a subset of another can be thought of as the process of specializing the credentials specification for the larger group. Thus, the smaller group's credentials are more specific than those of the larger group. Likewise, creating a larger group through the union of smaller groups generalizes the credentials specification of the smaller groups, thus resulting in less specific credentials for the larger group.

Rule #2: Deny first, allow later

A good security management principle is that of denying access to a resource unless access is explicitly granted. Thus, when specifying a network's security policy the first step must be to deny access to all target principals via rules that identify initiators via the broadest possible credentials. One can then grant access to each target principal solely to the group of principals to which access should be granted.

For example, to protect a set of host machines from access by all but a small set of principals, one can define a rule that denies access to these machines and whose initiator is denoted by ignore. A second rule allowing access can then be defined. It specifies the same target principal and, as the initiator, a credential specification that describes, in the narrowest possible manner, the principals being granted access. The ranking algorithm guarantees that the rule granting access ranks higher than the rule denying it.

It is crucial that the credential specification for the principals being granted the access privilege be as specific as possible, with all other principals being

denied access. This ensures that access is not inadvertently granted to non-privileged principals.

In general, the first policy rule in every protocol layer is one that denies
5 access to all and by all communicating entities (using ignore for both initiator and target principals) for all protocol actions (again using ignore).

Rule #3: Prerequisites are your friends, use them often and use them wisely

Prerequisite rules can play a critical role in the disambiguation of like-ranked
10 rules. Thus, prerequisites should be used whenever possible. In particular, prerequisites should be used in a way that targets each rule to the smallest set of principals possible, and that prevents the repetition of credentials within a set of related rules. For example, if an IP rule exists that defines communication between hosts in two subnets and we want to define a TCP
15 rule affecting the same set of hosts, we should define a TCP rule that takes the aforementioned IP rule as a prerequisite. In addition, the credentials used in the TCP rule should not include assertions that repeat what has already been established by the IP rule (*e.g.* the IP addresses of the relevant hosts). Instead the TCP rule credentials should specialize (if so desired) the
20 specification of the host credentials, *e.g.* limiting the host services covered by the rule (*i.e.* stating the IP ports of interest).

Rule #4: Make dispositions final, unless they are not

Immediate outcomes that produce a final disposition should be used
25 whenever possible. In other words, unless one knows that a rule at a given protocol layer may be overridden by a specific rule at a higher protocol layer,

the immediate outcome for the former rule should always produce the final disposition for the network event. This prevents a rule's outcome from being inadvertently subsumed by another protocol event.

- 5 In general, unless a rule is explicitly listed as a prerequisite rule for another rule higher in the protocol stack, its immediate outcome should produce the final disposition for the network event.

Rule #5: If you know the Agent, name it

- 10 If a policy rule only applies to communications within a specific network segment, restrict the rule's scope by specifying the Agent(s) reporting the protocol events for which this rule should be considered.

- By doing so, one eliminates that rule from being considered in events reported
15 by other Agents.

- Although the invention has been described in detail with reference to particular preferred embodiments, persons possessing ordinary skill in the art to which this invention pertains will appreciate that various modifications and
20 enhancements may be made without departing from the spirit and scope of the claims that follow.

CLAIMS

1. A method for a policy engine to organize policy rules in a way to effect
5 an efficient evaluation of a protocol event, said protocol event having an agent
descriptor, a protocol name, and a protocol action, said method comprising:

providing a first associative array having a first key and a first value,
wherein said first key corresponds to said agent descriptor and said first value
is a reference to a second associative array having a second key and a
10 second value;

providing said second associative array, wherein said second key
corresponds to said protocol name and said second value is a reference to a
third associative array having a third key and a third value;

providing said third associative array, wherein said third key
15 corresponds to said protocol action and said third value is a reference to a
fourth associative array having a fourth key and a fourth value; and

providing said fourth associative array, wherein said fourth key
corresponds to any of said rules and said fourth value is a rank number
associated with said any rule.

20

2. The method of Claim 1, further comprising ordering said rules in
decreasing order of rank number.

3. The method of Claim 2, further comprising incorporating constraints
25 into said ordering.

4. The method of Claim 2, further comprising ordering in lexical order rules having a same rank number.

5. The method of Claim 1, wherein said any rule is referenced by a plurality of said fourth associative arrays.

6. The method of Claim 1, wherein said rank number of said any rule is a relative value dependent on other rank numbers in said fourth associative array.

10

7. An apparatus for a policy engine to organize policy rules in a way to effect an efficient evaluation of a protocol event, said protocol event having an agent descriptor, a protocol name, and a protocol action, said apparatus comprising:

15 means for providing a first associative array having a first key and a first value, wherein said first key corresponds to said agent descriptor and said first value is a reference to a second associative array having a second key and a second value;

means for providing said second associative array, wherein said second key corresponds to said protocol name and said second value is a reference to a third associative array having a third key and a third value;

20 means for providing said third associative array, wherein said third key corresponds to said protocol action and said third value is a reference to a fourth associative array having a fourth key and a fourth value; and

means for providing said fourth associative array, wherein said fourth key corresponds to any of said rules and said fourth value is a rank number associated with said any rule.

5 8. The apparatus of Claim 7, further comprising means for ordering said rules in decreasing order of rank number.

9. The apparatus of Claim 8, further comprising means for incorporating constraints into said ordering.

10

10. The apparatus of Claim 8, further comprising means for ordering in lexical order rules having a same rank number.

11. The apparatus of Claim 7, wherein said any rule is referenced by a plurality of said fourth associative arrays.

15

12. The apparatus of Claim 7, wherein said rank number of said any rule is a relative value dependent on other rank numbers in said fourth associative array.

20

1/37

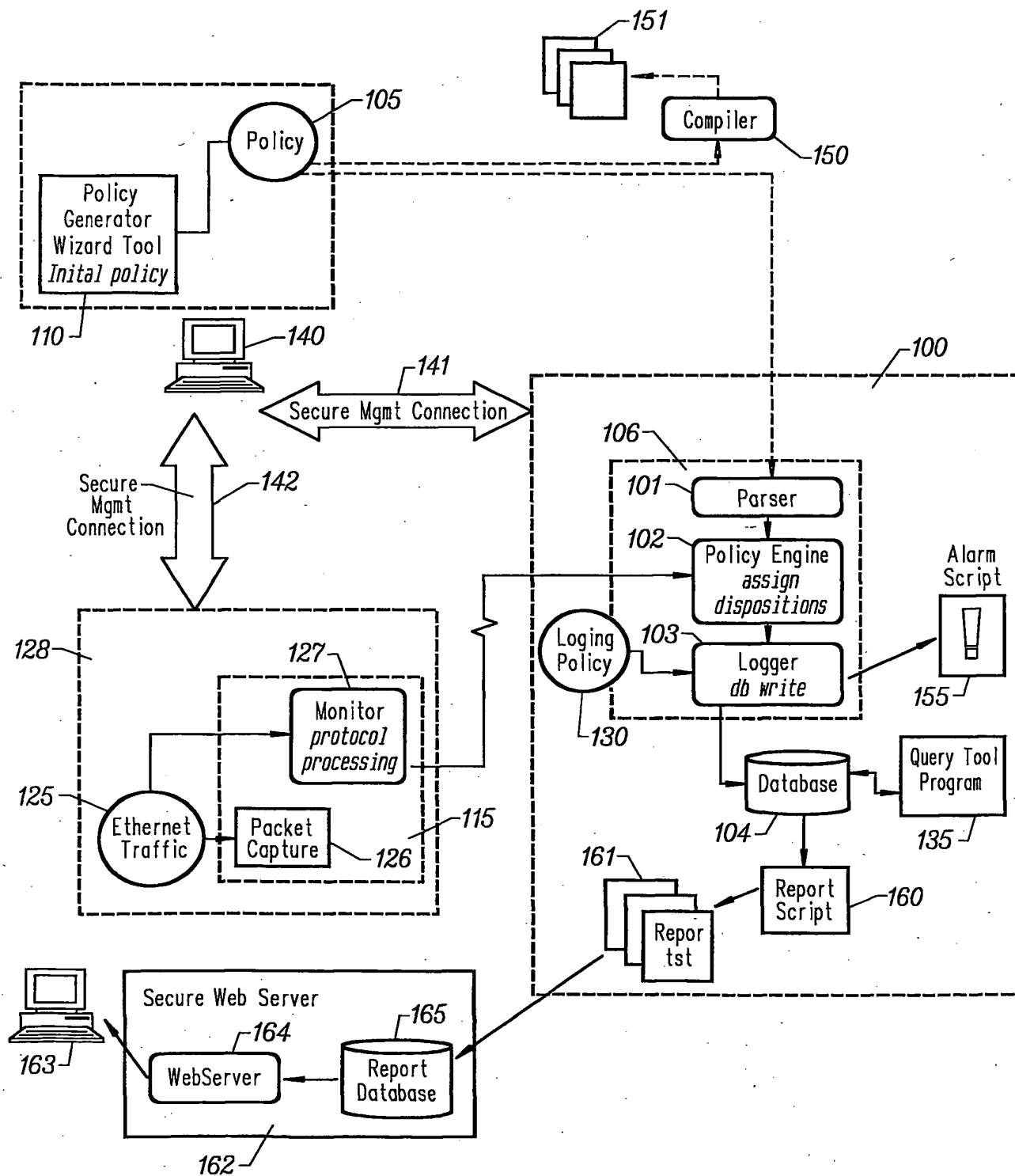


FIG. 1A

2/37

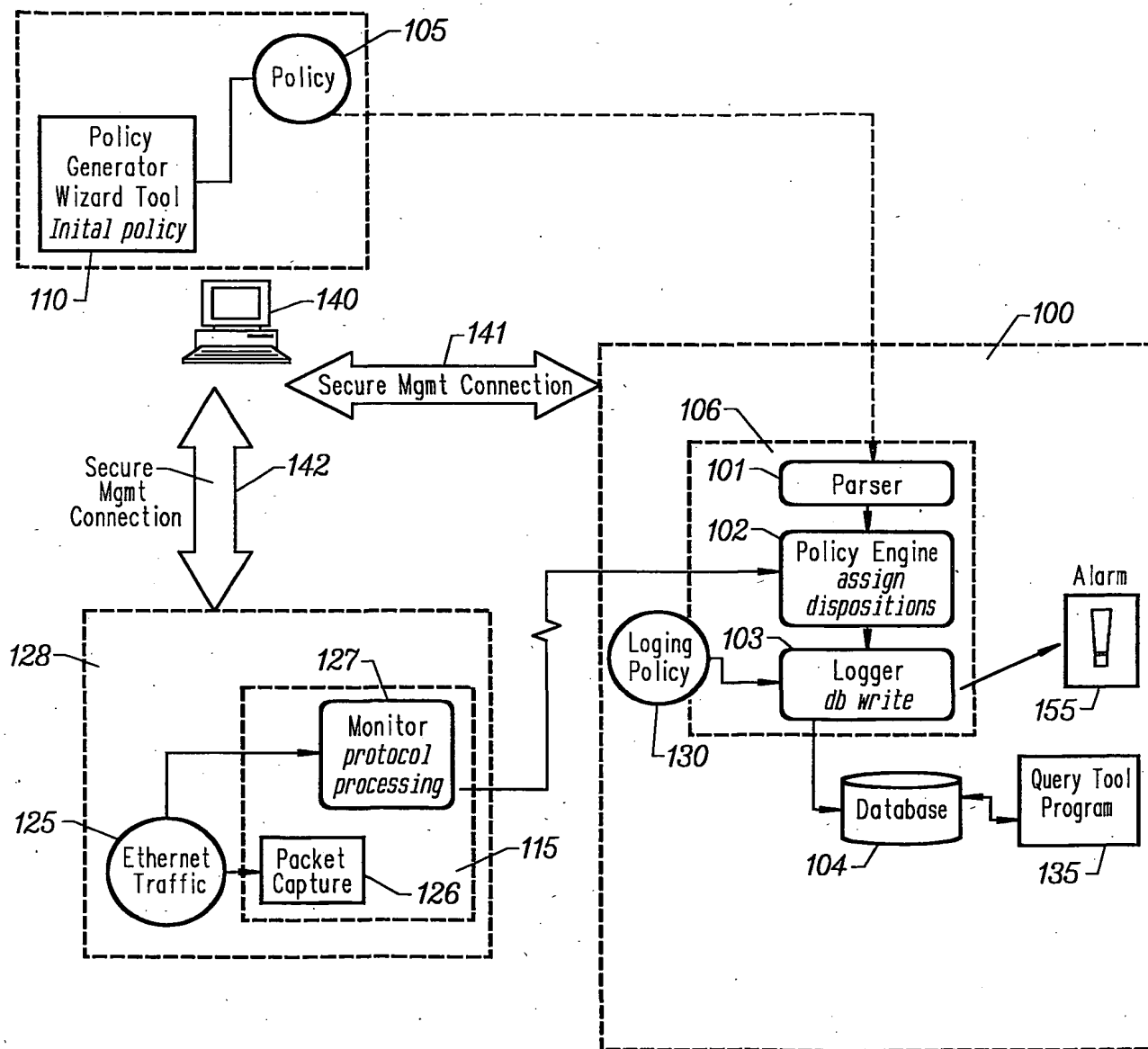
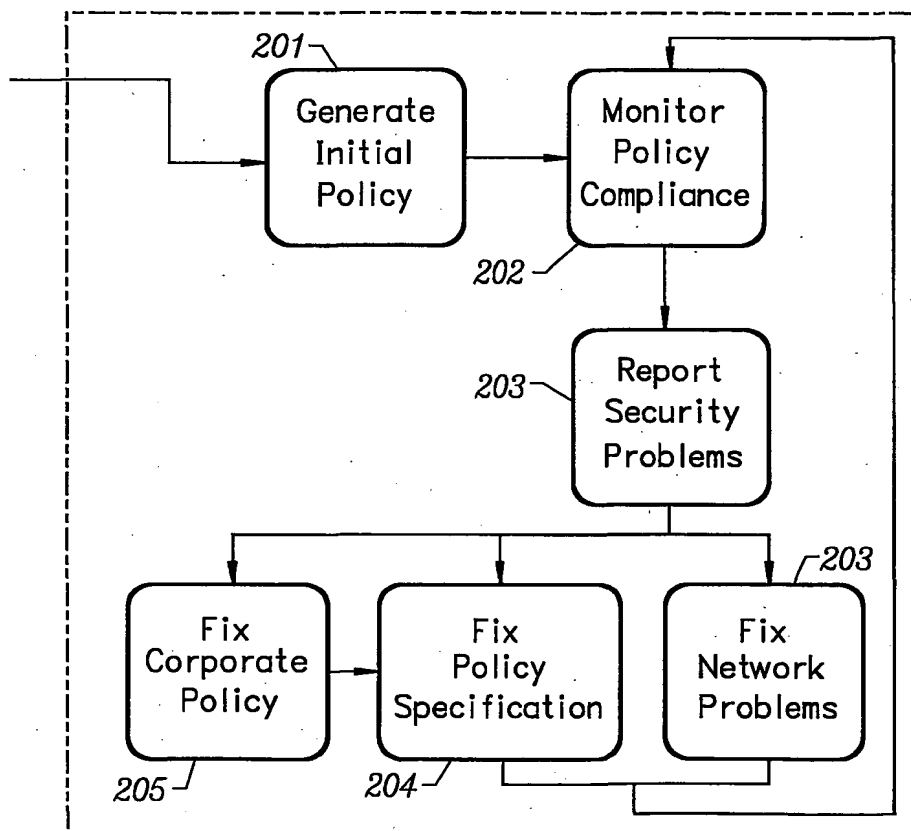


FIG. 1B

3/37

*FIG. 2*

4/37

301

K Policy Generator

File Help

☐ ☐ ☐ ☐

Community	Policy Domains	Rules	Service
Name	Includes	Excludes	Description
Inside_Nodes	10.0.0.0/8		The Hosts in out Intranet <input type="checkbox"/>
Outside_Nodes		Inside_Nodes	All hosts in the Intranet <input type="checkbox"/>

☐ New ☐ X Delete Find Uses

FIG. 3

5/37

K Policy Processor

Output Director

Browse

Output File 402

Process Policy 401

Close

FIG. 4A

6/37

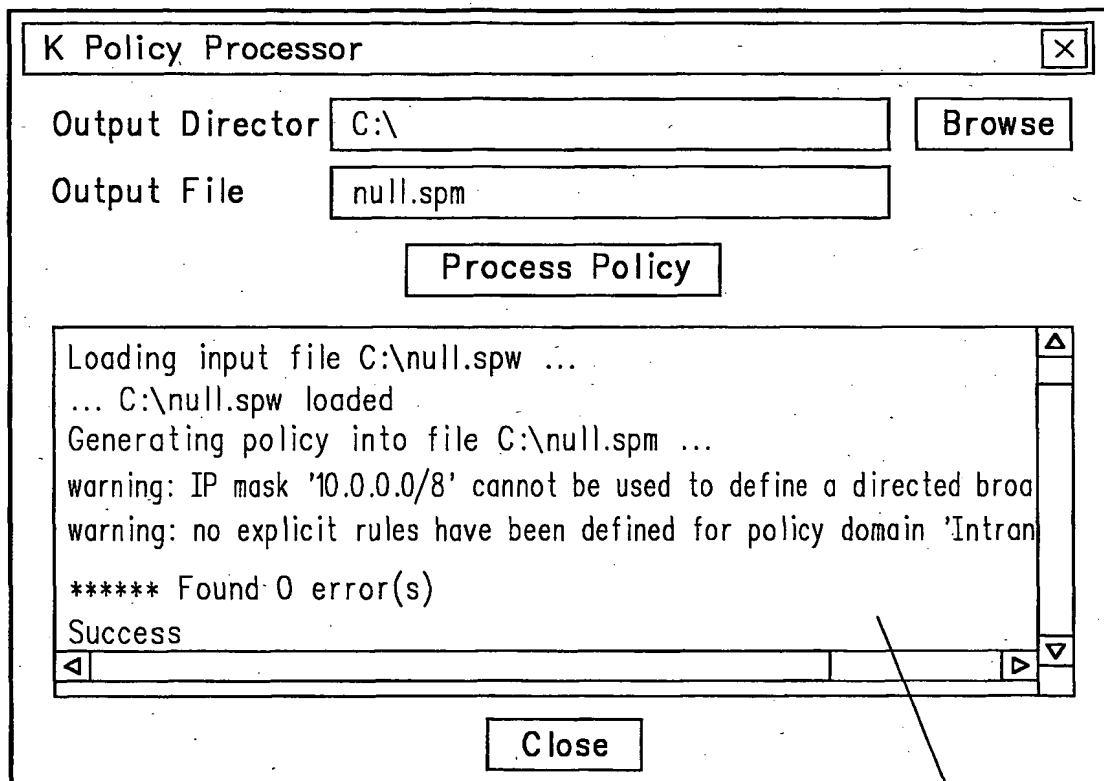


FIG. 4B

7/37

SPM: Argument Selector Dialog

Monitor configuration

Input dump file: C:\qs.dmp 501 Browse

Policy: C:\null.spm 502 Browse

Monitorign Point: INTRANET_MONITOR 503
(comma separated)

Monitor Logging Options

Execution Run Comment:

ODBC name: sybase 504

DB Username: policy 505

DB Password: ***** 506 ☒ Save Password [insecure]

Output Options

☐ Output to console:

☒ Output to file: C:\output.txt Browse

Run 507

Exit

Advanced

Help

Progress

nPkts

100%

0%

FIG. 5

8/37

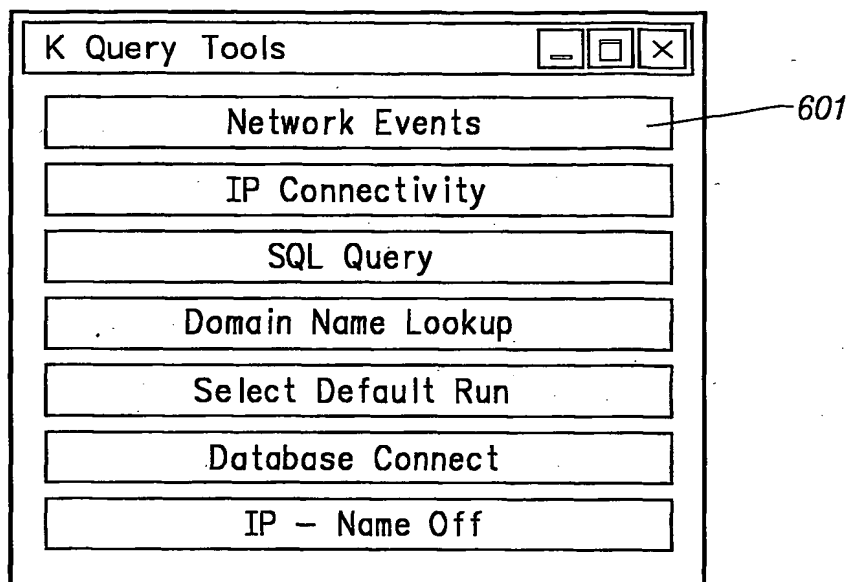


FIG. 6

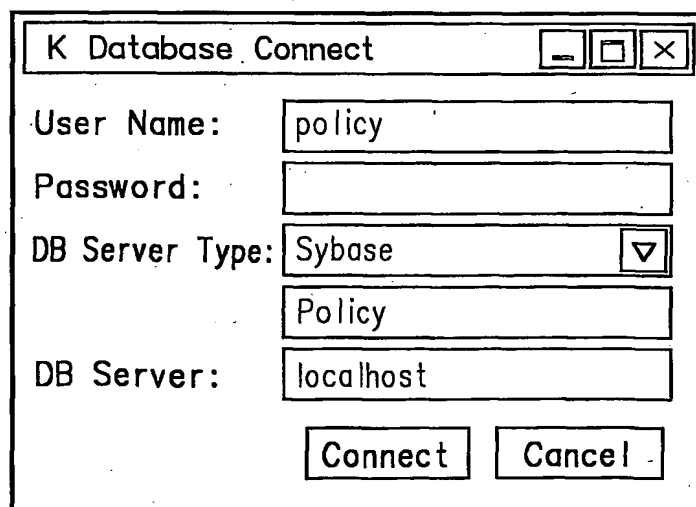


FIG. 7

9/37

K Rule View		X	
Execution Run:		1999-10-01 14:30:20.0 C:\.bmp	
Final Rule Name:		<Any Rule>	
Disposition Name:		<Any Disposition>	
Disposition Codes:		<input type="checkbox"/> Access Denied <input type="checkbox"/> Auth Violation <input type="checkbox"/> Security Attack <input type="checkbox"/> Security QOS <input type="checkbox"/> Policy Error <input type="checkbox"/> OK	
Disposition Severity:		<input type="checkbox"/> Critical <input type="checkbox"/> High <input type="checkbox"/> Medium <input type="checkbox"/> Monitor <input type="checkbox"/> Warning <input type="checkbox"/> Information <input type="checkbox"/> <none>	
		Query	
Rows		Done Edit SQL Copy Row Copy Deep	

FIG. 8

10/37

K Rule View	
Execution Run:	1999-10-01 14:30:20.0 C:\.bmp
Final Rule Name:	<Any Rule>
Disposition Name:	<Any Disposition>
Disposition Codes: <input type="checkbox"/> Access Denied <input type="checkbox"/> Auth Violation <input type="checkbox"/> Security Attack <input type="checkbox"/> Security QOS <input type="checkbox"/> Policy Error <input type="checkbox"/> OK	
Disposition Severity: <input type="checkbox"/> Critical <input type="checkbox"/> High <input type="checkbox"/> Medium <input type="checkbox"/> Monitor <input type="checkbox"/> Warning <input type="checkbox"/> Information <input type="checkbox"/> <none>	
Query	

Rule Name	Disposition Name	Initiator IP	Init Name	Target IP	Targ Name	Targ Service
Udp_Deny	Udp_Access_Denied	10.5.63.143	vg-143.securify.com	10.5.63.6	dude.securify.com	domain
Http_Deny	Http_Access_Denied	10.5.63.143	vg-143.securify.com	208.178.27.198		http
Http_Deny	Http_Access_Denied	10.5.63.143	vg-143.securify.com	208.178.27.201		http
Http_Deny	Http_Access_Denied	10.5.63.143	vg-143.securify.com	208.178.27.198		http
Udp_Deny	Udp_Access_Denied	10.5.63.143	vg-143.securify.com	10.5.63.6	dude.securify.com	domain
Udp_Deny	Udp_Access_Denied	10.5.63.143	vg-143.securify.com	10.5.63.6	dude.securify.com	domain
Http_Deny	Http_Access_Denied	10.5.63.143	vg-143.securify.com	204.71.200.68	www3.yahoo.com	http
Udp_Deny	Udp_Access_Denied	10.5.63.143	vg-143.securify.com	10.5.63.6	dude.securify.com	domain
Http_Deny	Http_Access_Denied	10.5.63.143	vg-143.securify.com	10.5.63.97	kabale.securify.com	http
Tcp_Missed_Connections	Warn_Missed_Tcp_Connect	10.5.63.143	vg-143.securify.com	10.5.63.24	fred.securify.com	netbios-ssn

Rows 10	Done	Edit SQL	Copy Row	Copy Deep
---------	------	----------	----------	-----------

FIG. 9

11/37

FIG. 10A

K Policy Generator				<input type="button" value="Min"/> <input type="button" value="Max"/> <input type="button" value="X"/>	
File Help					
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>					
Community	Policy Domains	Rules	Service		
Select Policy Domain Policy Domain: <input type="text" value="Intranet"/>					
Identify New or Existing Rule in Intranet Rule Name: <input type="text" value="Internal_Dns"/> <input type="checkbox"/> New <input type="checkbox"/> Delete					
Add Elements to Internal_Dns Description: <input type="text"/> <input type="button" value="Set"/>					
Initiators: <div> <input checked="" type="checkbox"/> Intranet Inside_Nodes ... Firewall ... Outside_Nodes </div> <input type="button" value="Add Selected"/>		Services: <div> AUTH BOOTP_CLIENT BOOTP_SERVER DNS FINGER </div> <input type="button" value="Add Selected"/>		Targets: <div> <input checked="" type="checkbox"/> Intranet Inside_Nodes ... Firewall ... Outside_Nodes </div> <input type="button" value="Add Selected"/>	
Rule Contents for Internet#_Dns					
Initiators: <div> <input type="text" value="Any"/> </div> <input type="button" value="Add Selected"/>		Services: <div> <input type="text" value="Any"/> </div> <input type="button" value="Add Selected"/>		Targets: <div> <input type="text" value="Any"/> </div> <input type="button" value="Add Selected"/>	

12/37

K Policy Generator			
File Help			
<div> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> </div>			
Community	Policy Domains	Rules	Service
<div> <div>Select Policy Domain</div> <div>Policy Domain: <input type="text" value="Intranet"/></div> </div>			
<div> <div>Identify New or Existing Rule in Intranet</div> <div> <div>Rule Name: <input type="text" value="Internal_Dns"/></div> <div><input type="checkbox"/> New</div> <div><input checked="" type="checkbox"/> Delete</div> </div> </div>			
<div> <div>Add Elements to Internal_Dns</div> <div>Description:</div> <div>Allow DNS to be served from any internal host</div> <div>Set</div> </div>			
<div> <div>Initiators:</div> <div> <div>== Intranet ==</div> <div>Inside_Nodes</div> <div>... Firewall ...</div> <div>Outside_Nodes</div> </div> </div>		<div> <div>Services:</div> <div> <div>AUTH</div> <div>BOOTP_CLIENT</div> <div>BOOTP_SERVER</div> <div>DNS</div> <div>FINGER</div> </div> </div>	
Add Selected		Add Selected	
<div> <div>Targets:</div> <div> <div>== Intranet ==</div> <div>Inside_Nodes</div> <div>... Firewall ...</div> <div>Outside_Nodes</div> </div> </div>		Add Selected	
<div> <div>Rule Contents for Internet# Dns</div> <div> <div>Initiators:</div> <div>Inside_Nodes</div> </div> </div>			
Add Selected		Add Selected	
<div> <div>Services:</div> <div>DNS</div> </div>		<div> <div>Targets:</div> <div>Inside_Nodes</div> </div>	
Add Selected		Add Selected	

FIG. 10B

13/37

K Policy Generator

File

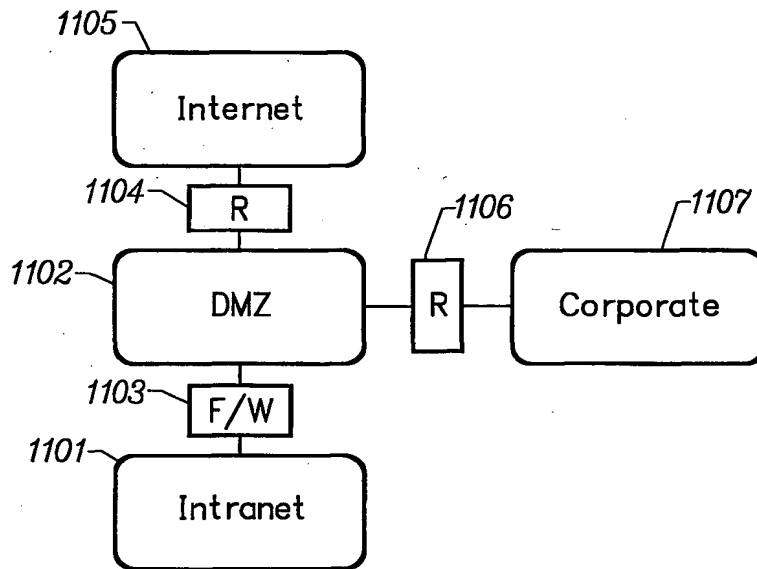
Help

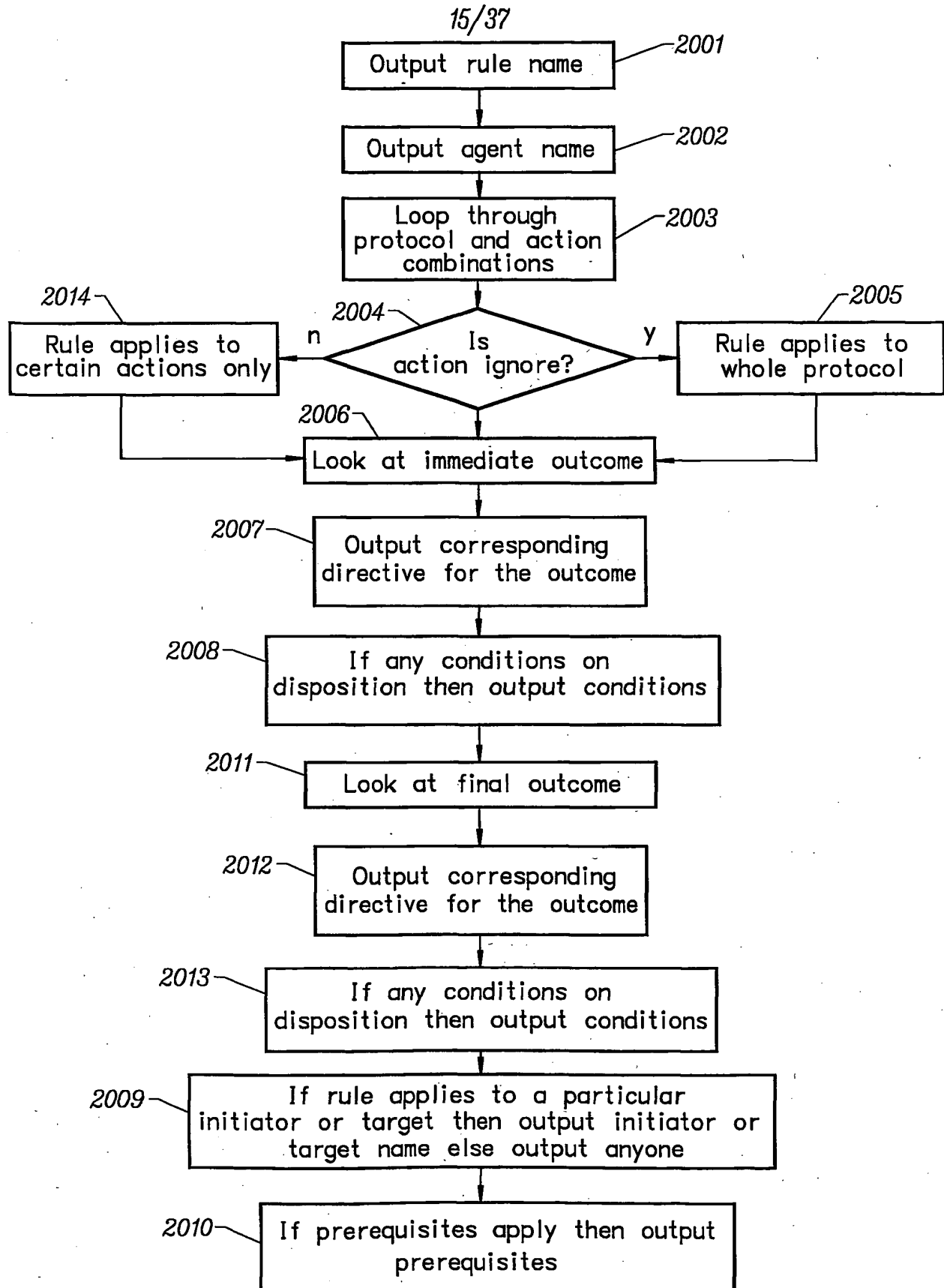
Community	Policy Domains	Rules	Service	
Name		Includes	Excludes	Description
Inside_Nodes	10.0.0.0/8			The Hosts in out Intranet
Outside_Nodes			Inside_Nodes	All hosts in the Intranet

☐ New
 ☒ Delete
 Find Uses

FIG. 10C

14/37

*FIG. 11*

*FIG. 12*

16/37

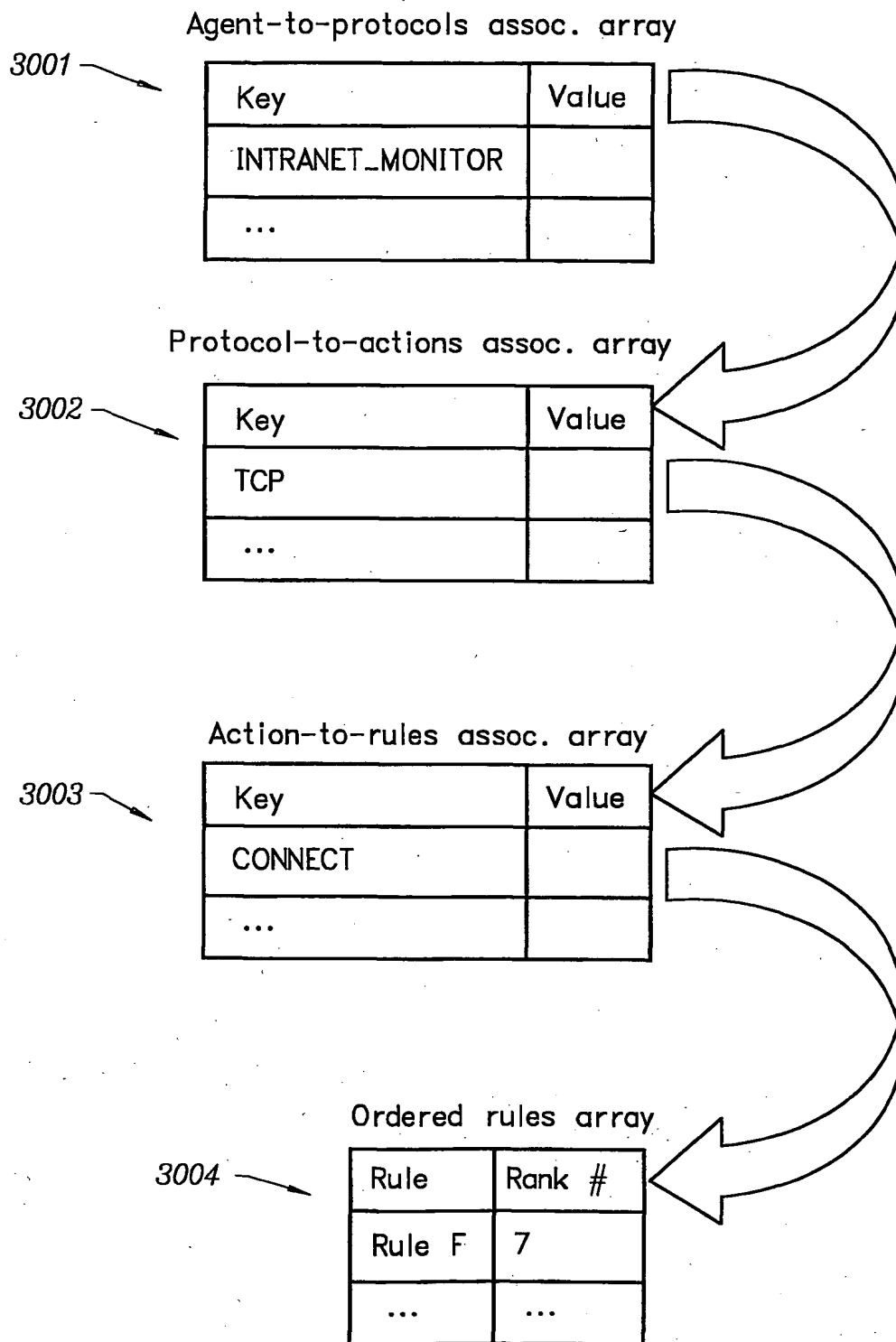


FIG. 13

17/37

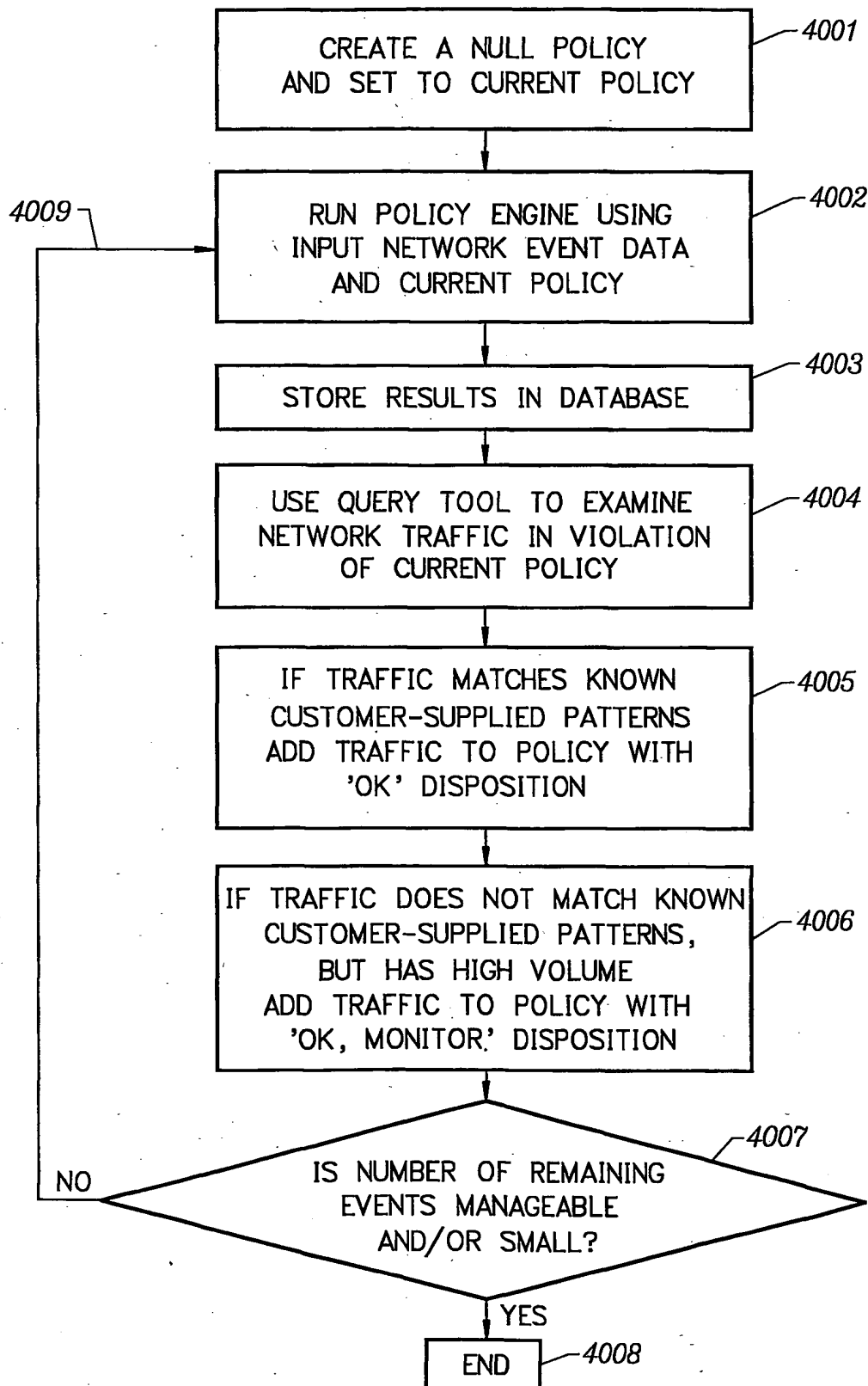


FIG. 14

18/37

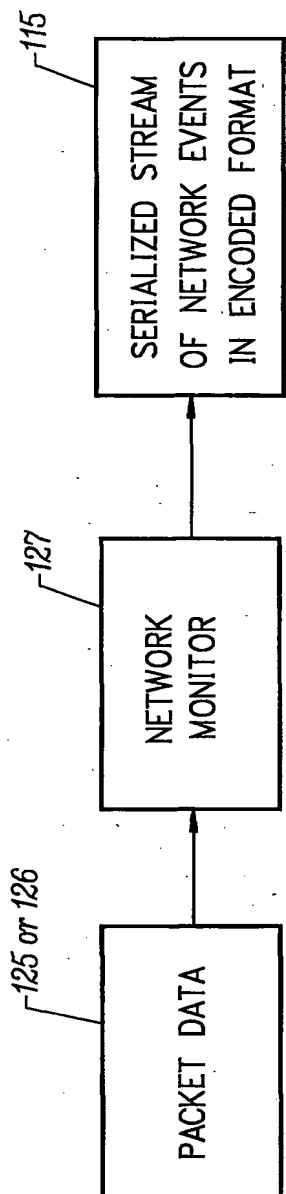


FIG. 15

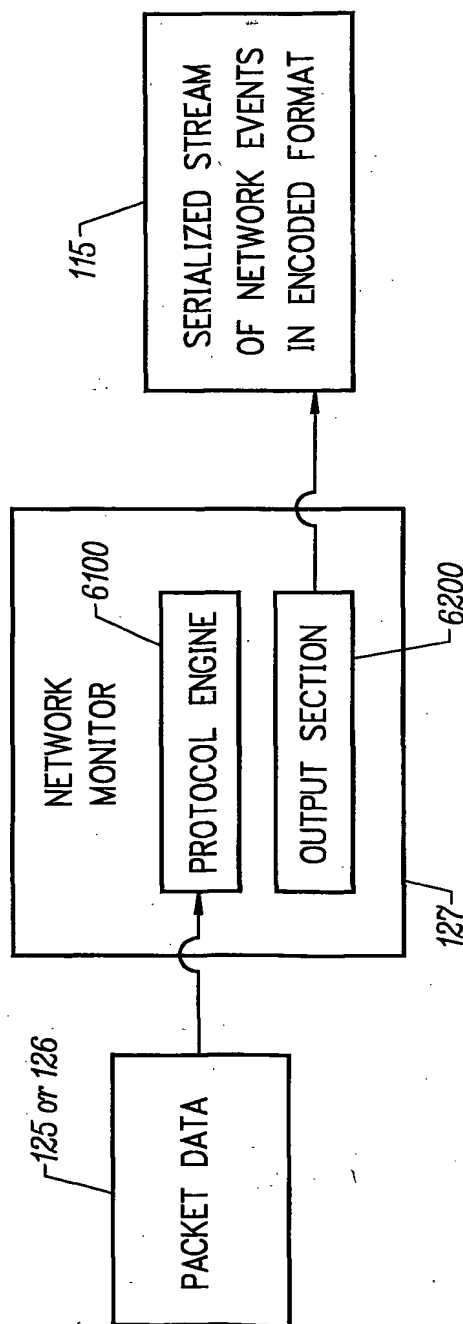


FIG. 16

19/37

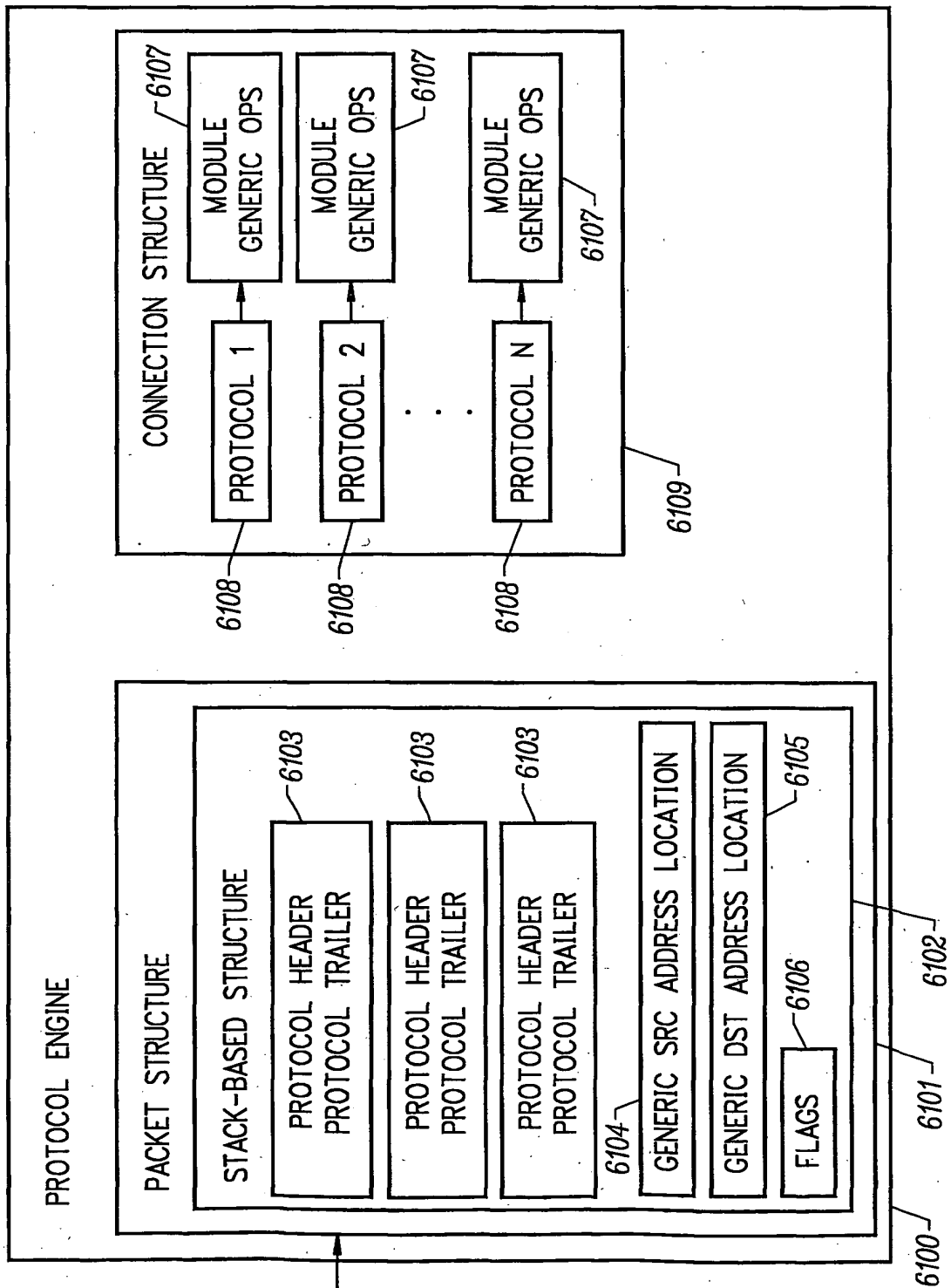


FIG. 17

20/37

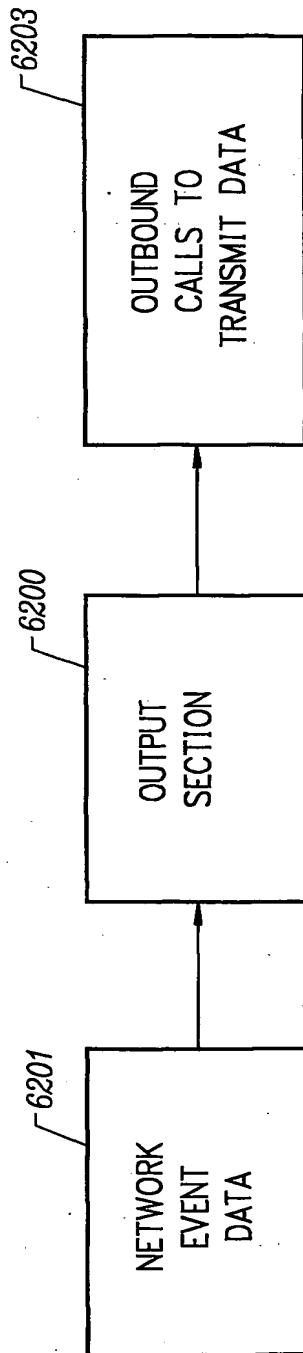


FIG. 18

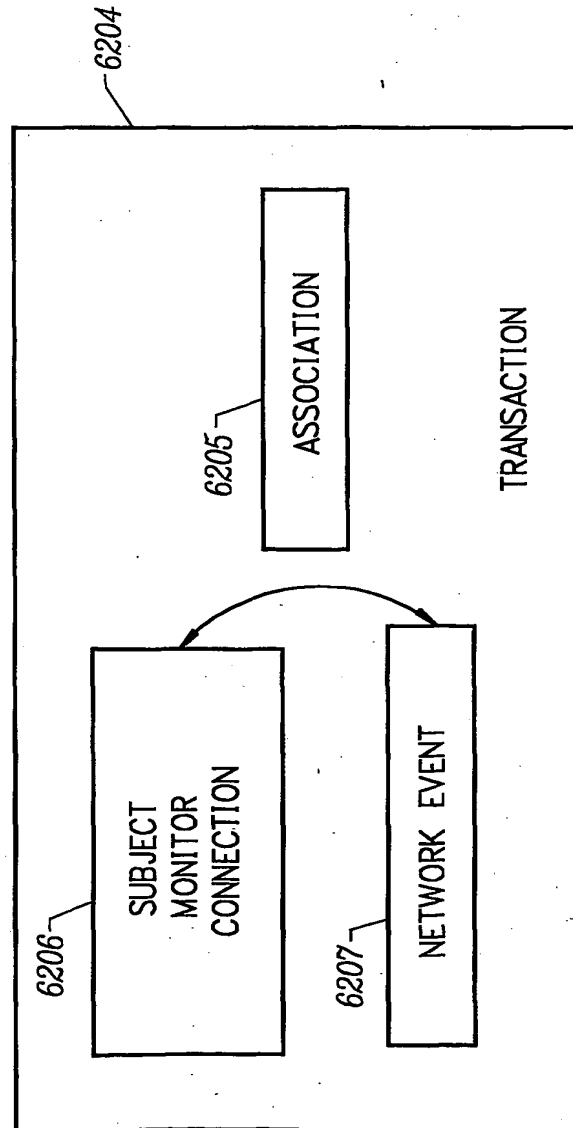
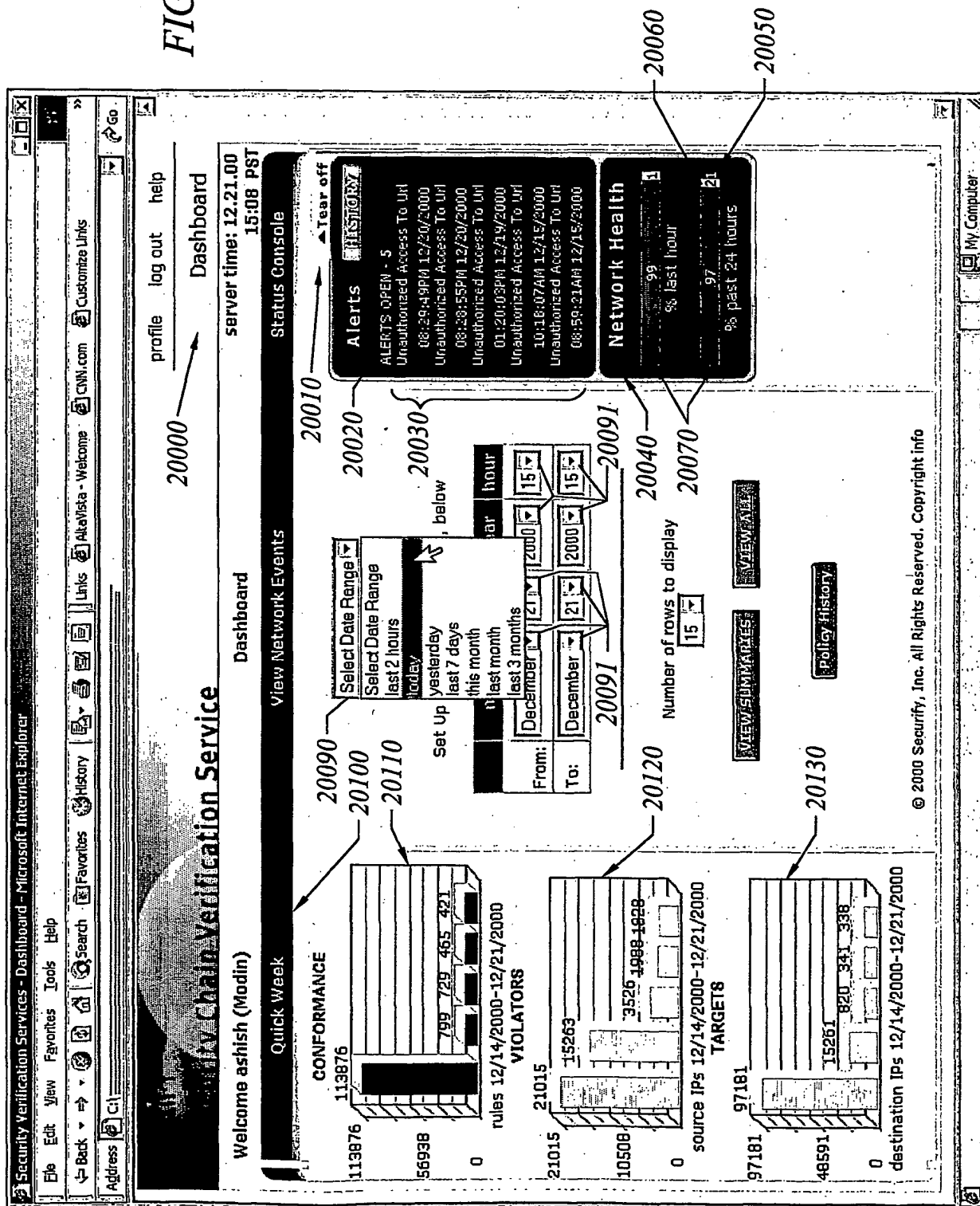


FIG. 19

21/37

FIG. 20



22/37

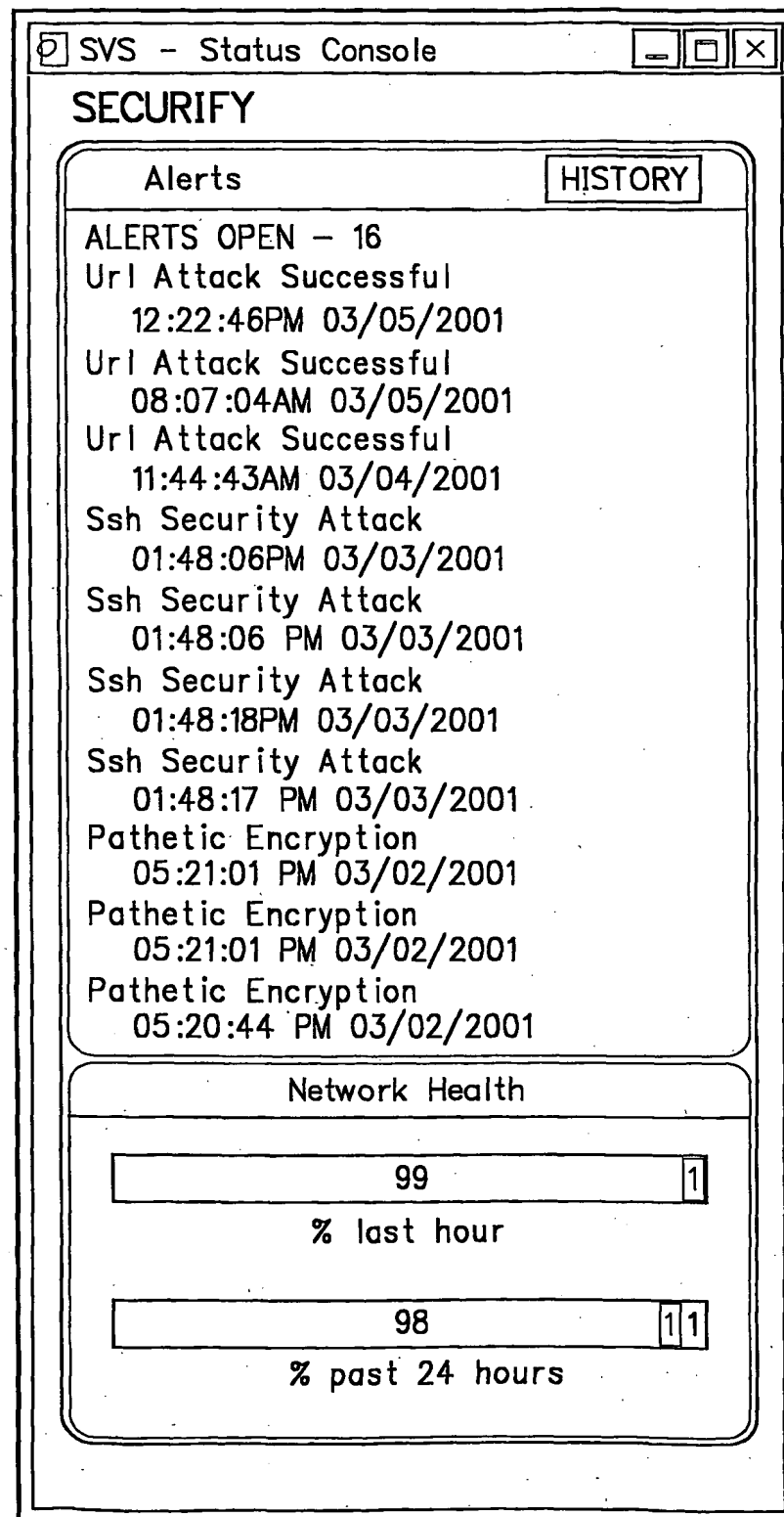
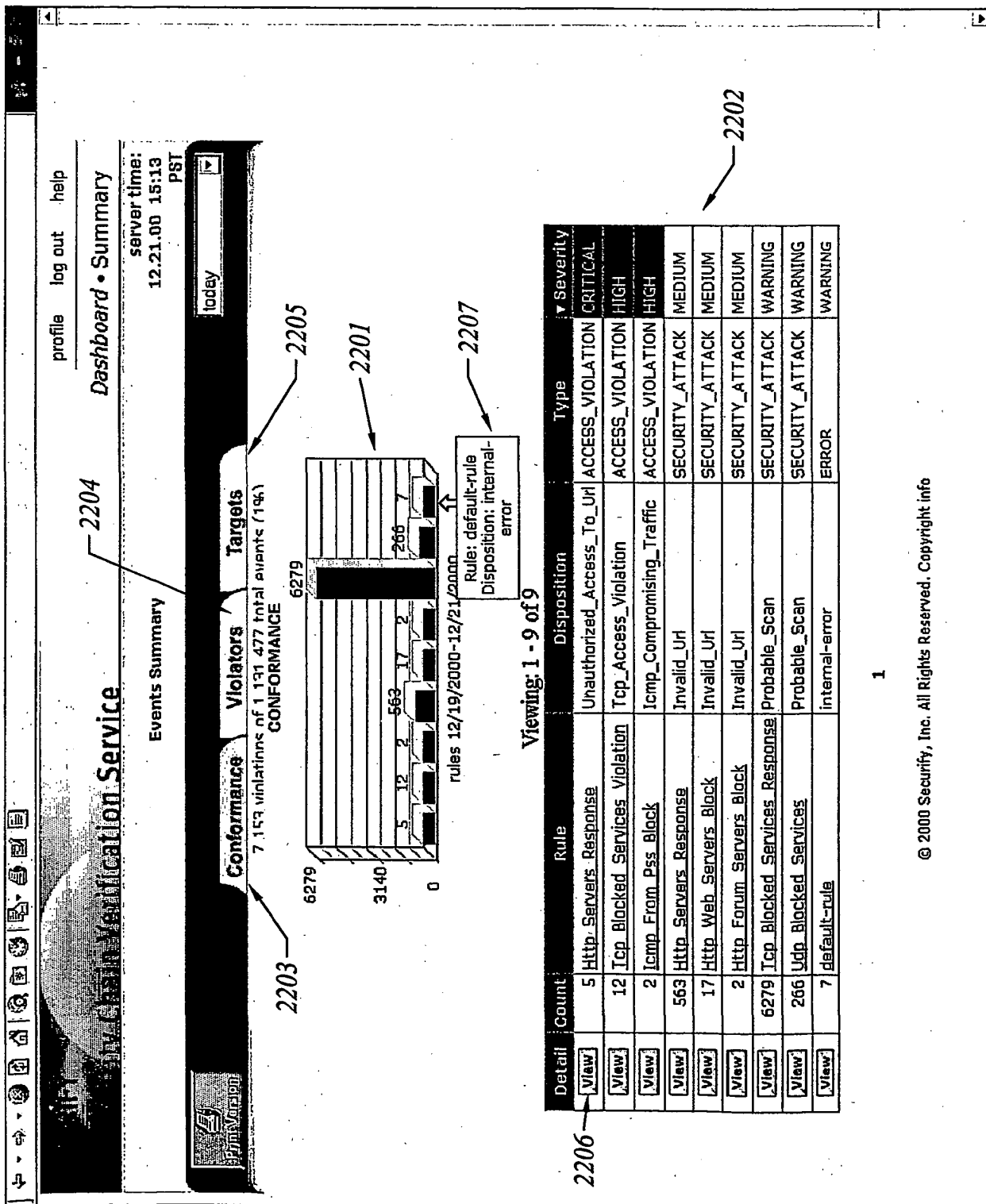


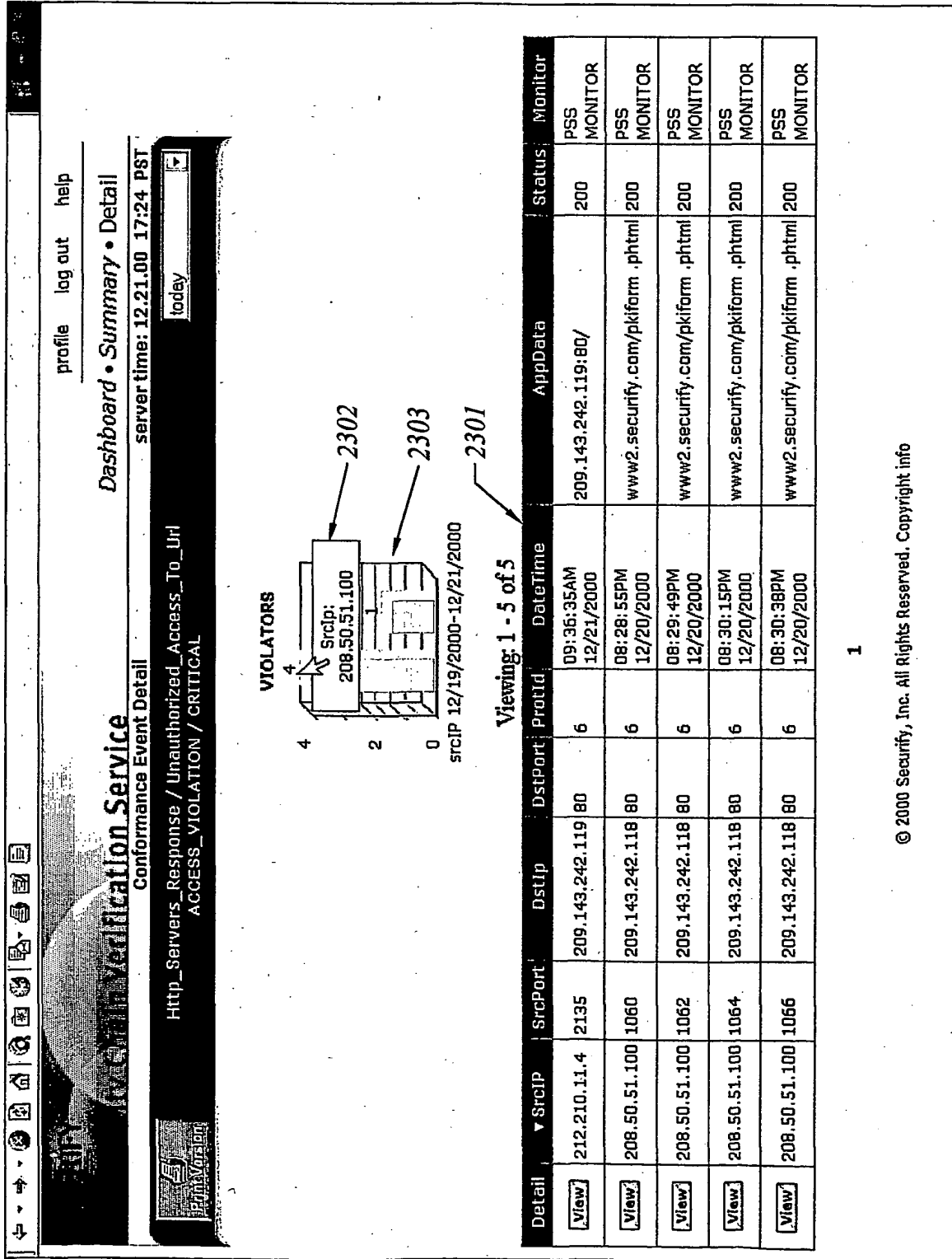
FIG. 21

FIG. 22



24/37

FIG. 23



25/37

FIG. 24

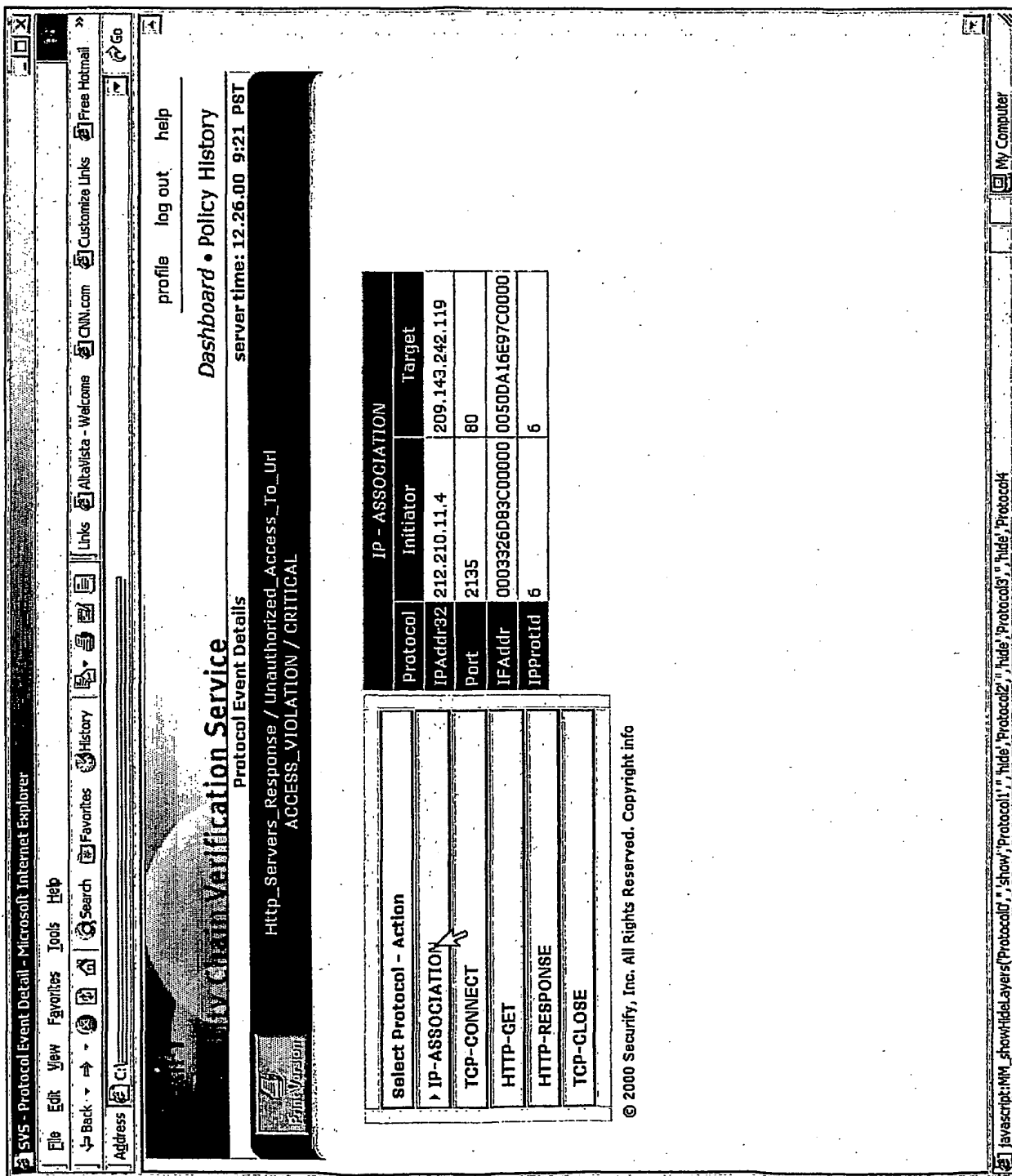


FIG. 25

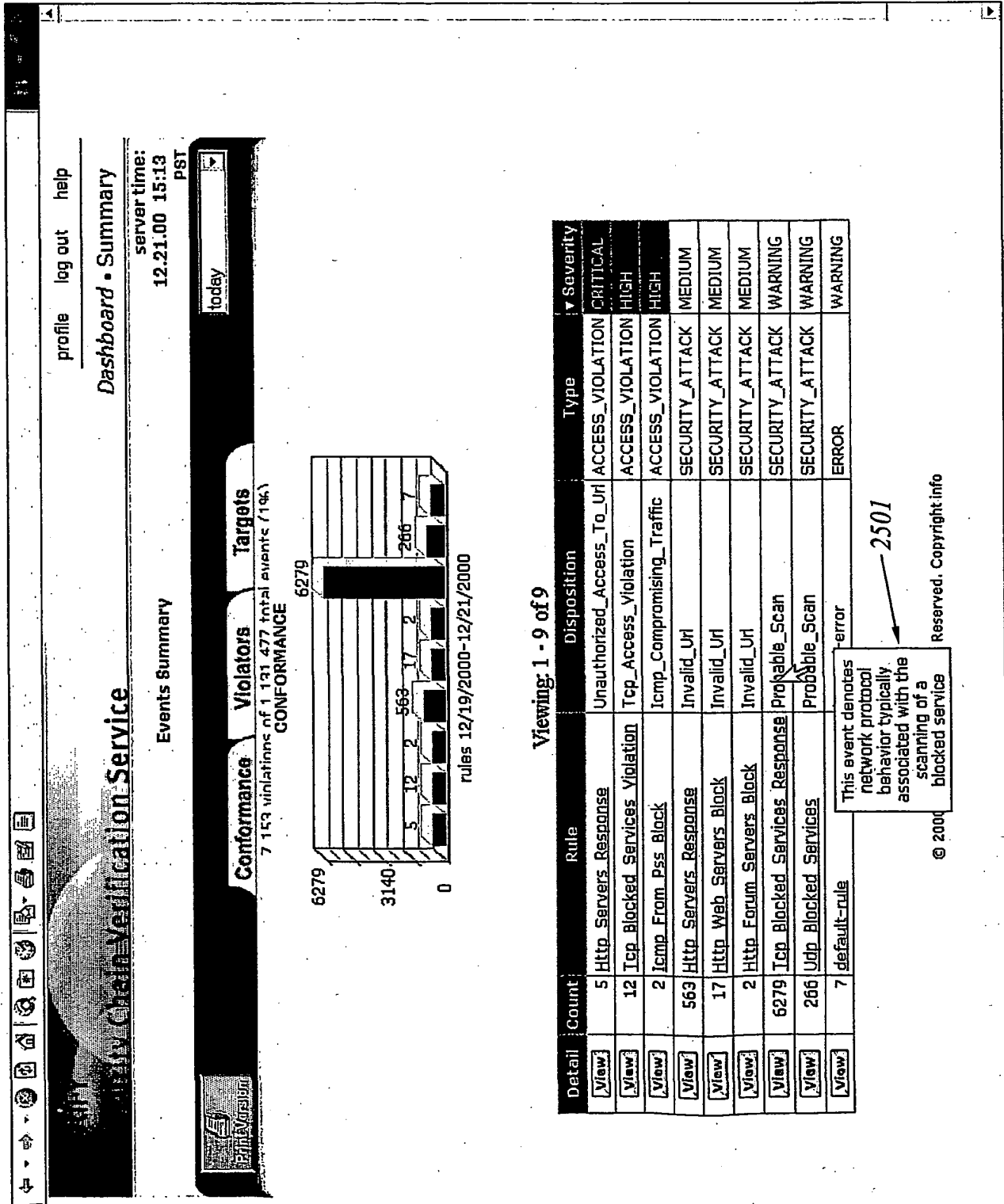
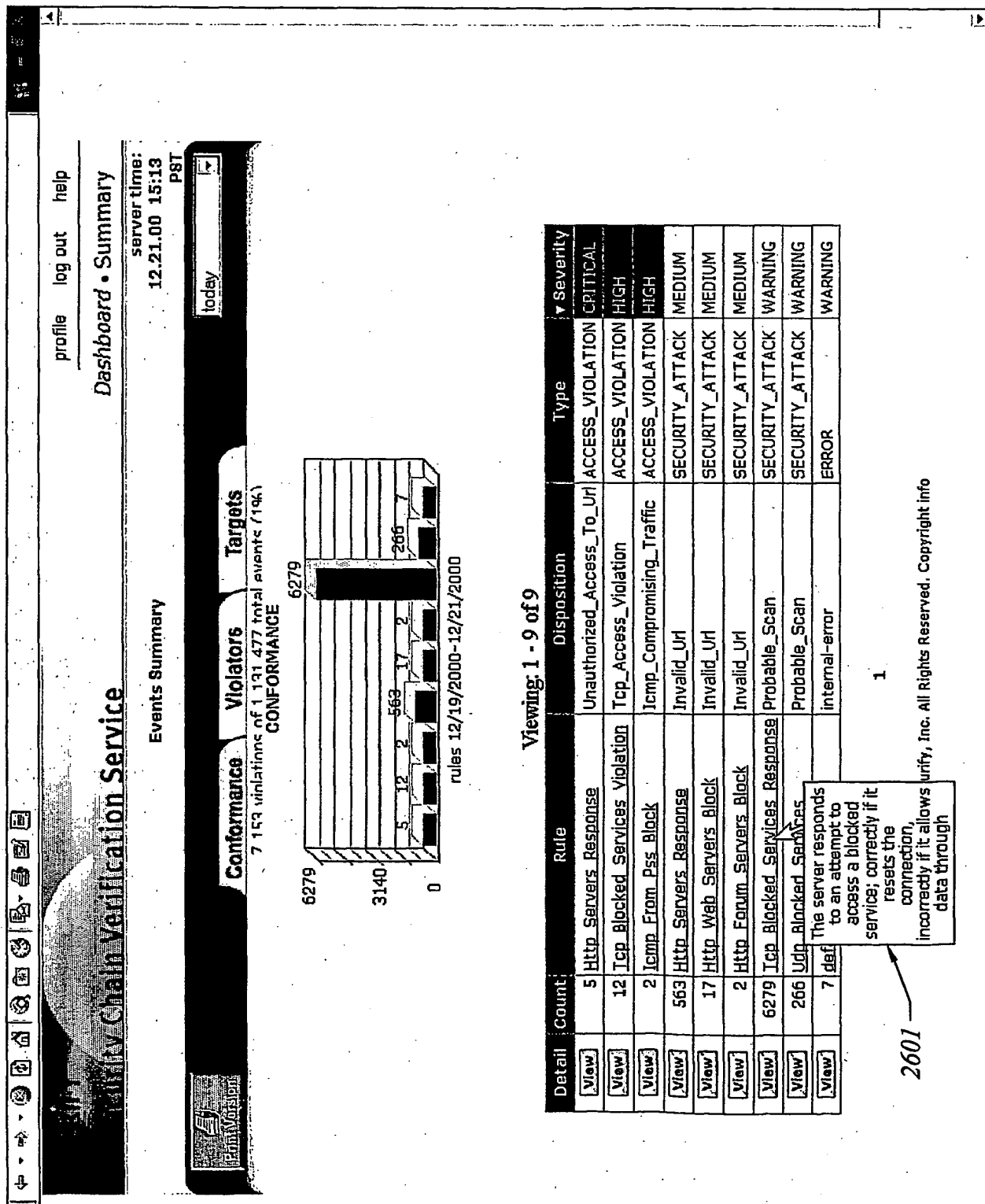
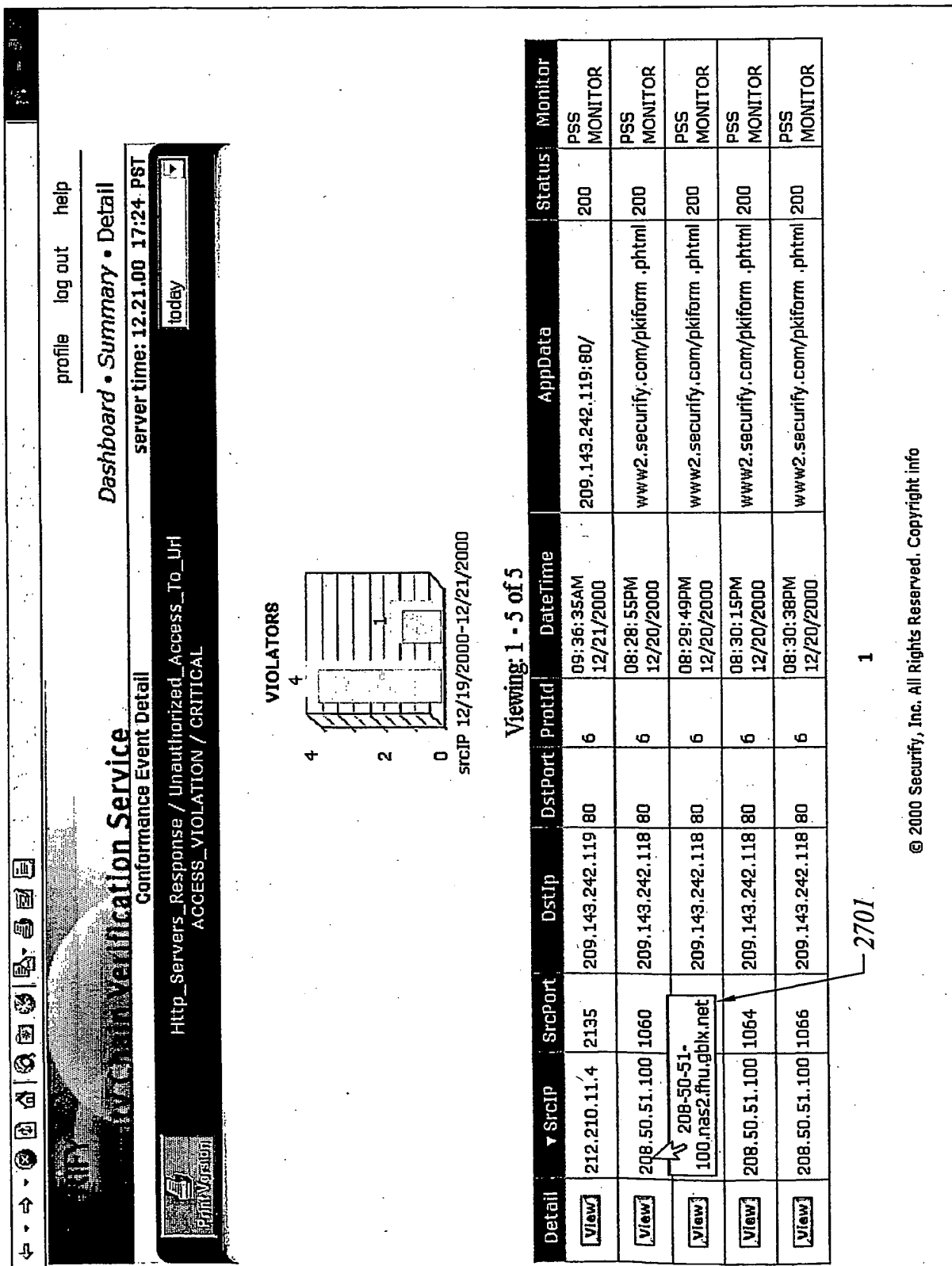


FIG. 26

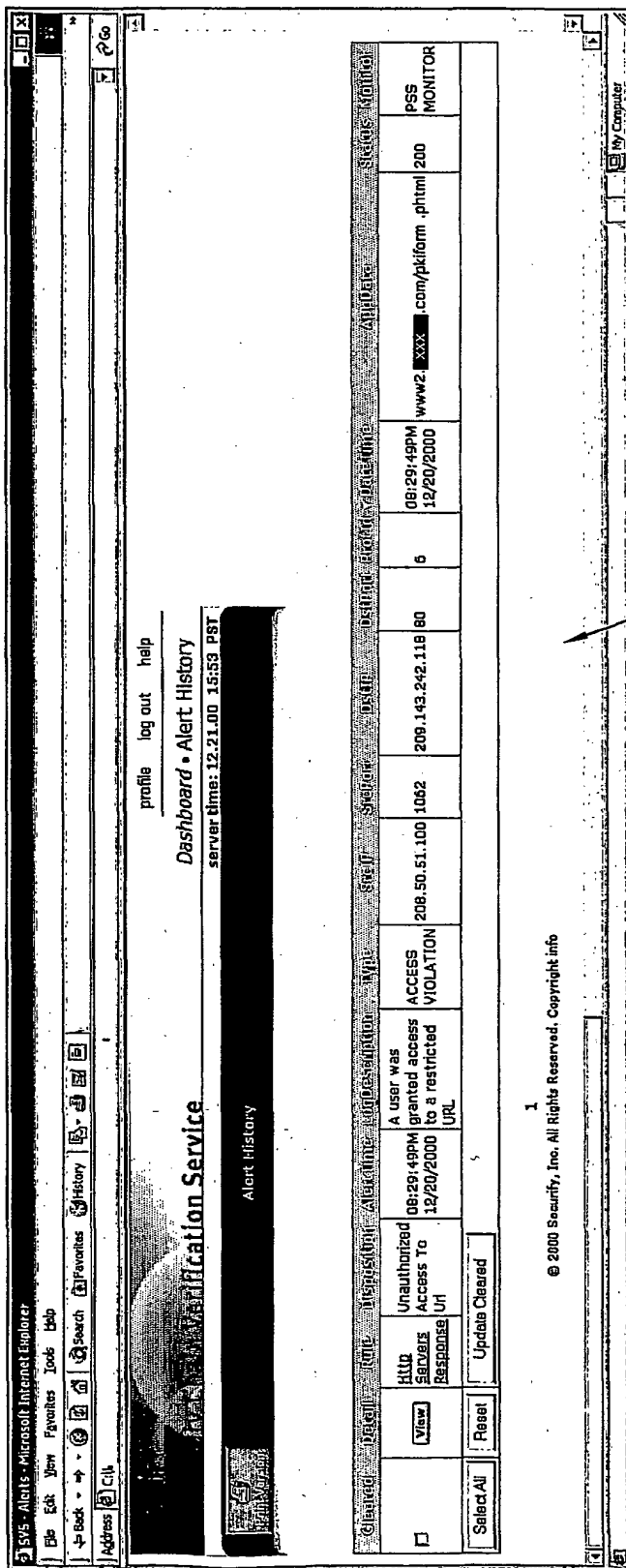


28/37

FIG. 27



29/37



2801

FIG. 28

30/37

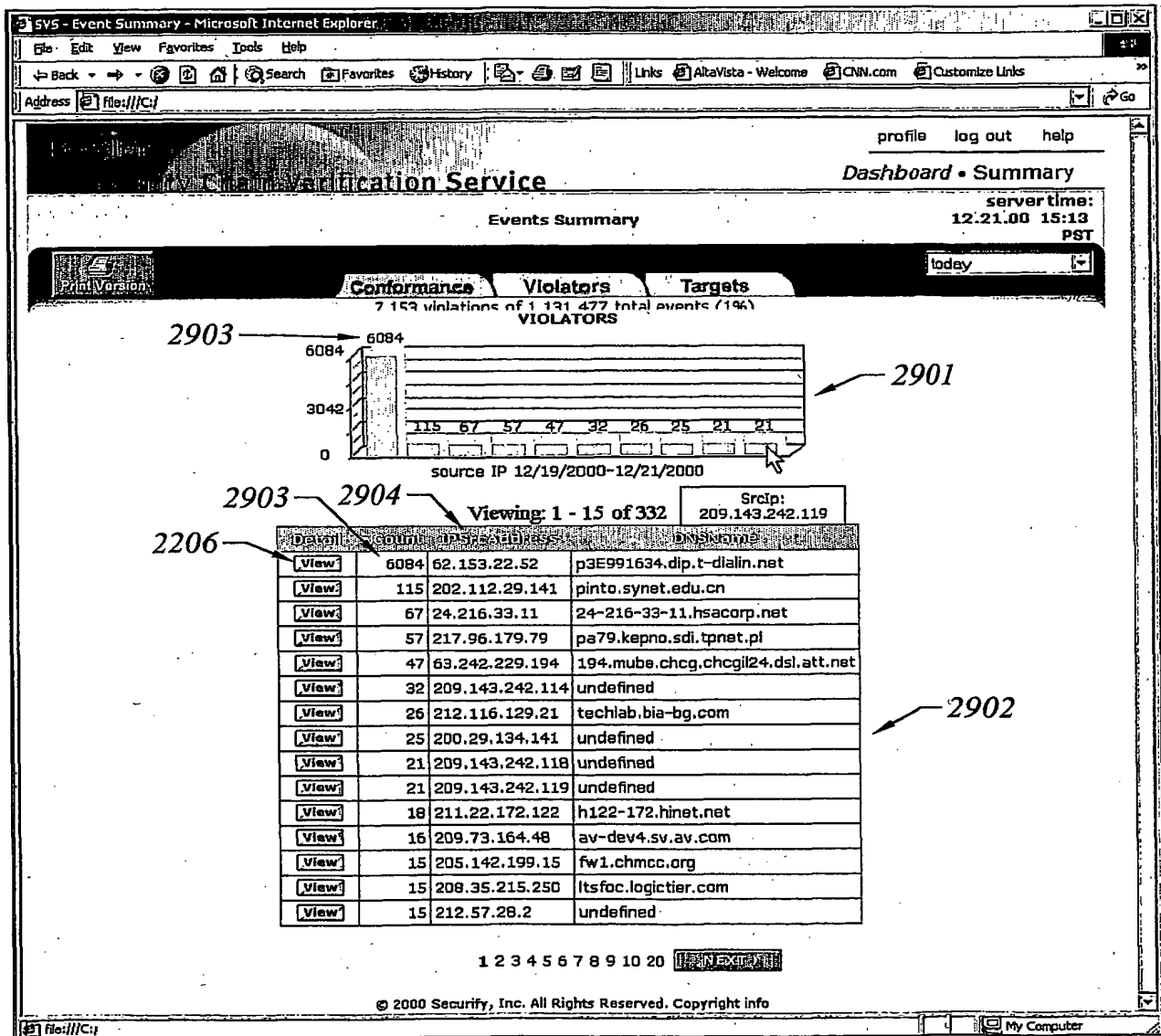


FIG. 29

31/37

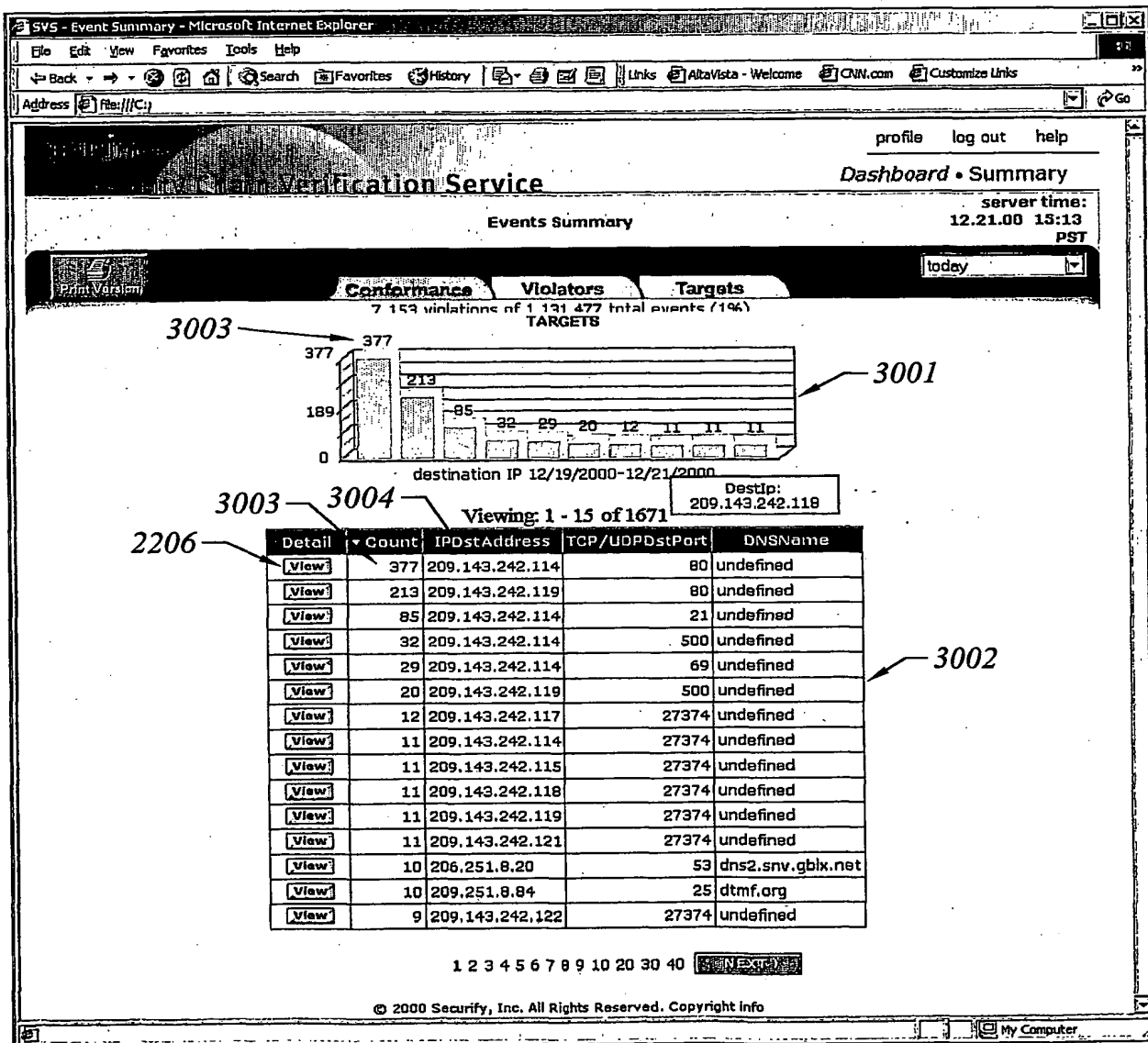


FIG. 30

32/37

Advanced Search - Microsoft Internet Explorer

Address <https://> Go

SECURIFY

Advanced Search

Filter results by One or All of the following:

Protocol

Rule

or
(regular expression in Rule)

Disposition

or
(regular expression in Disposition)

Source IP

Target IP

TargetPort

Monitor(s)
INTRANET_LOCAL_MONITOR
INTRANET_MONITOR
PARTNER_A_MONITOR

© 2000, 2001 Securify, Inc. All Rights Reserved.
Copyright info

3101

3104

3105

3106

3106

3102

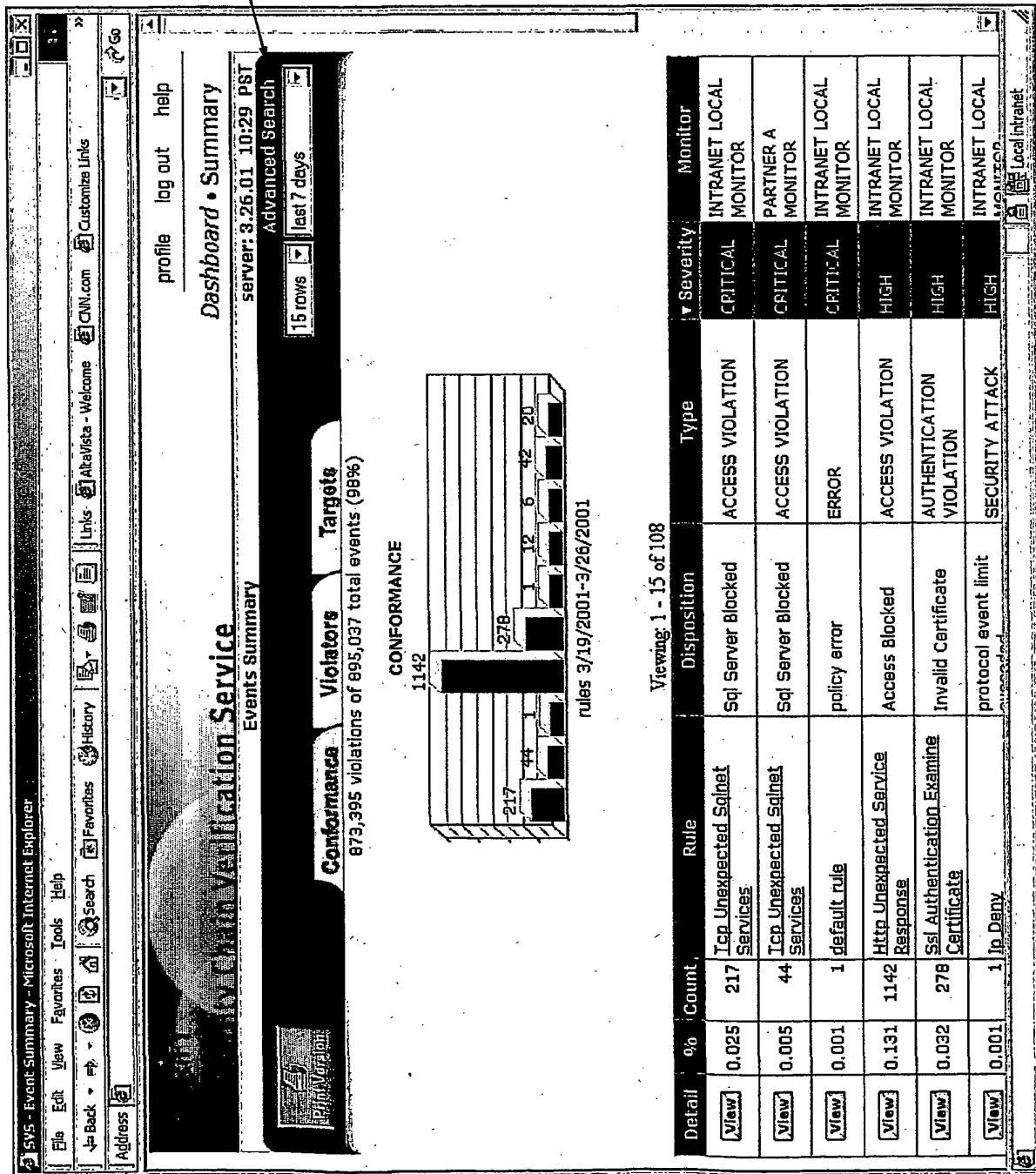
3103

3100

FIG. 31

33/37

FIG. 32



34/37

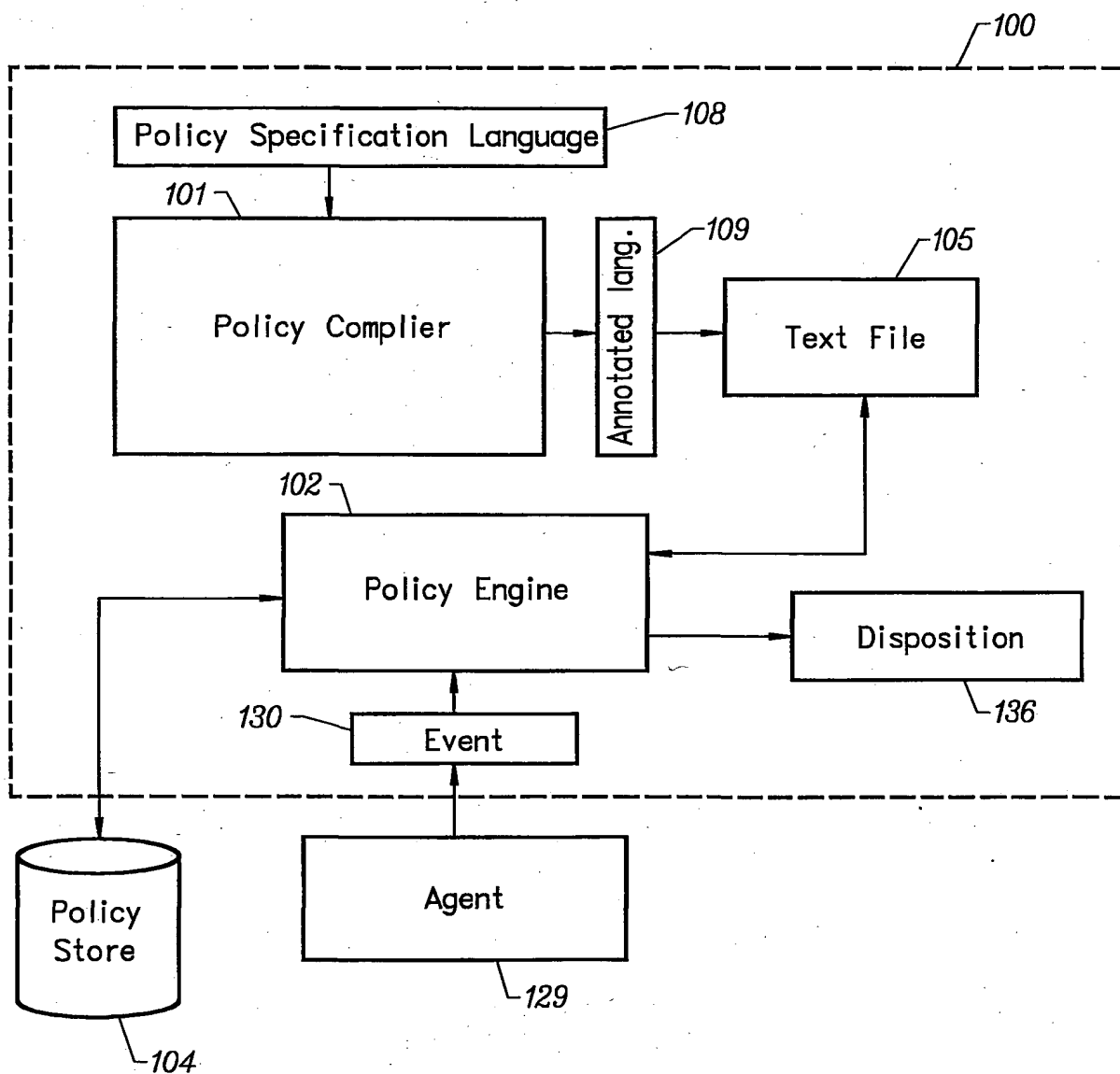


FIG. 33

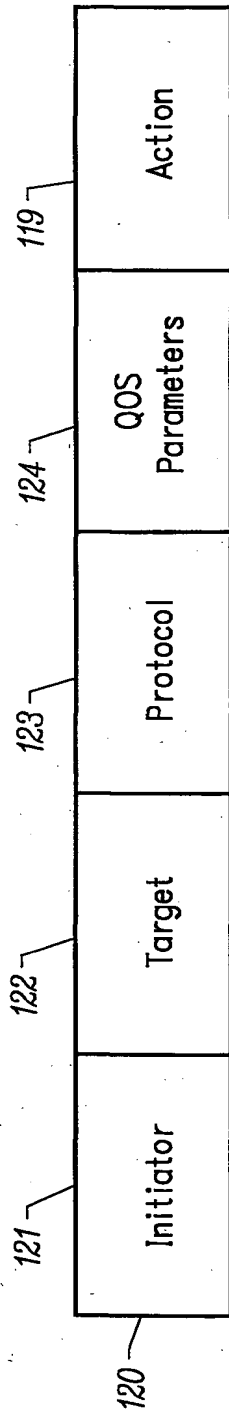


FIG. 34

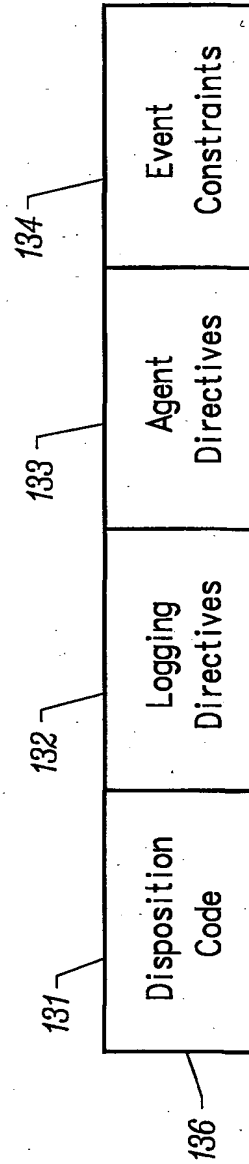


FIG. 35

36/37

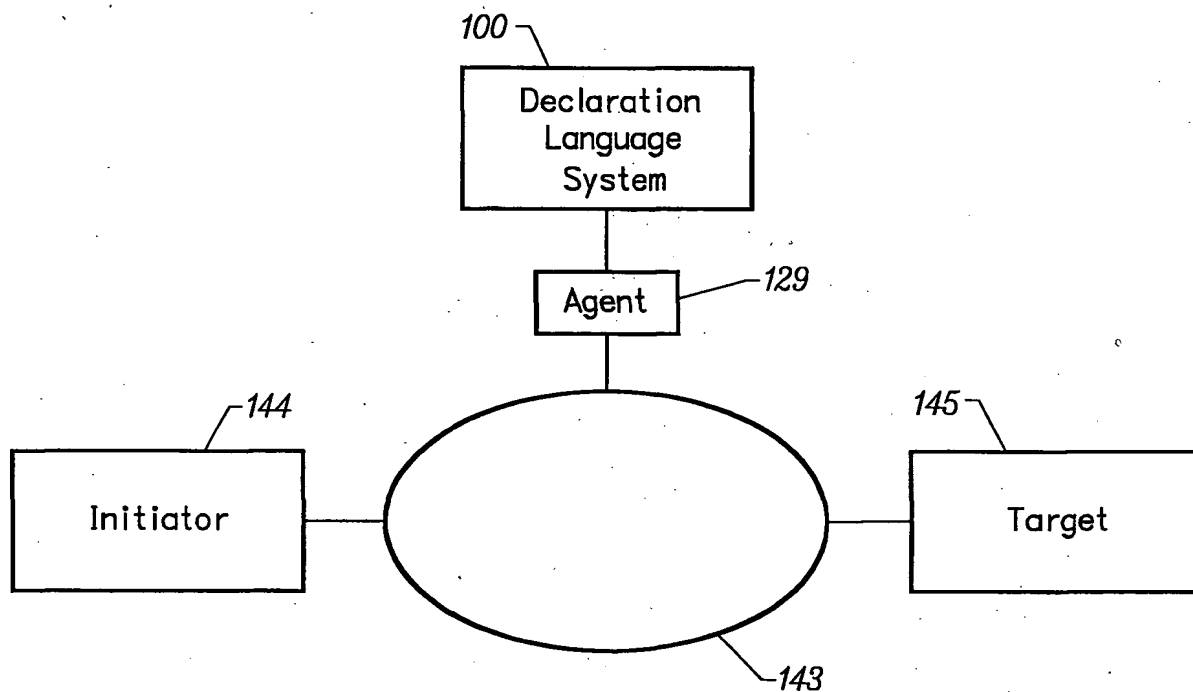


FIG. 36

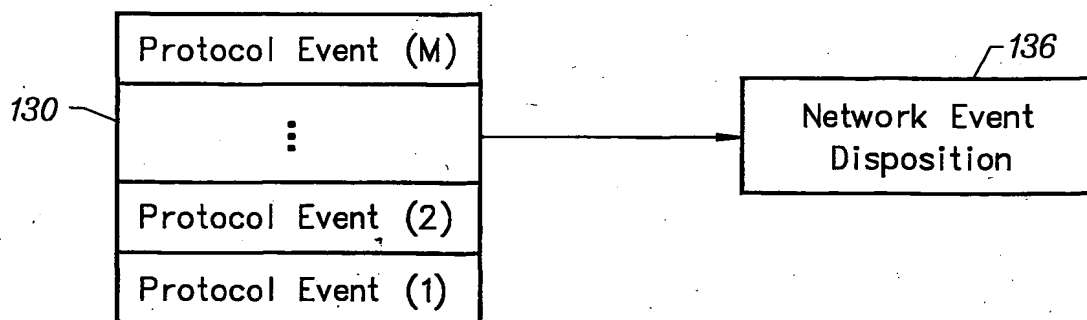


FIG. 37A

37/37

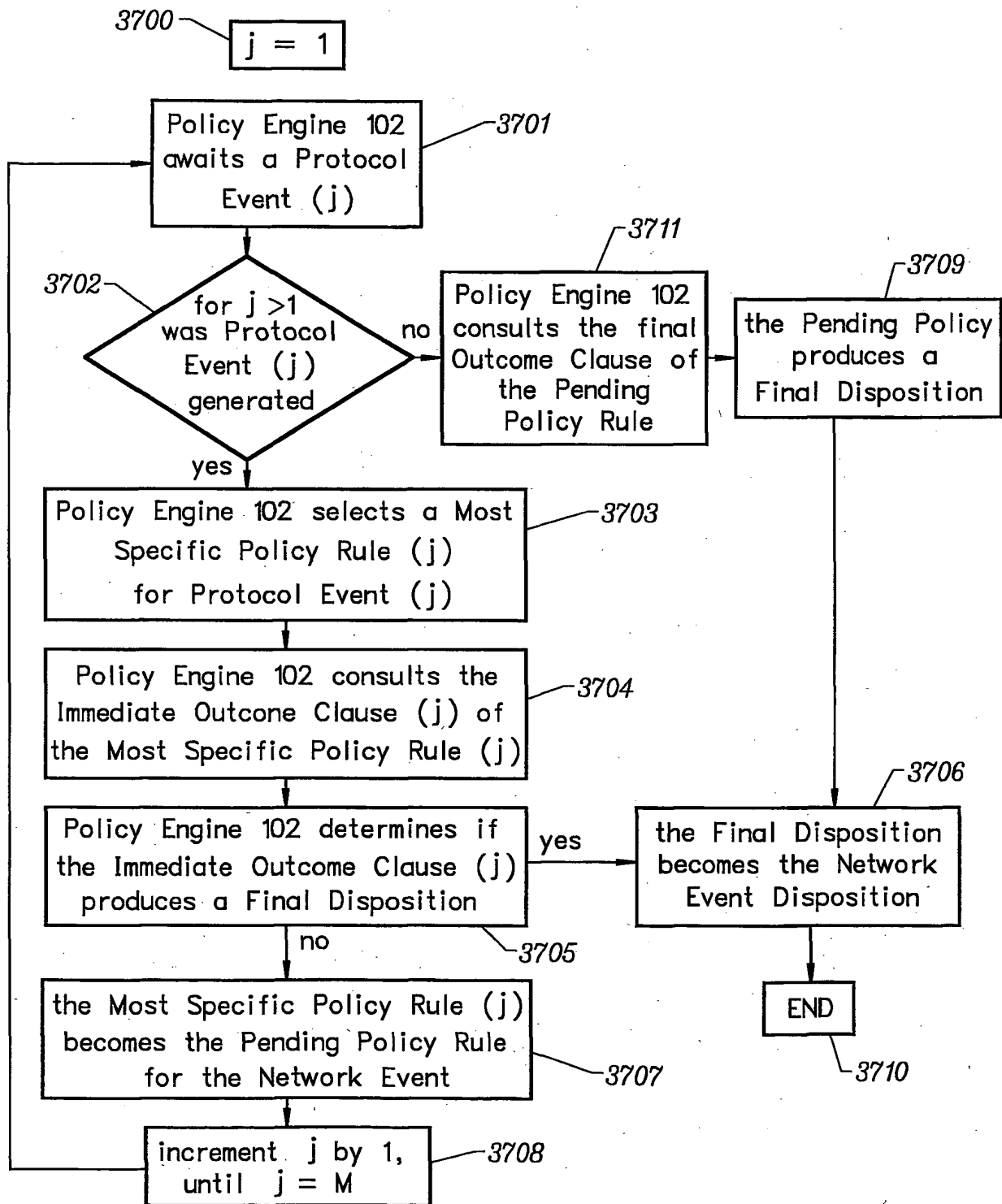


FIG. 37B

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.